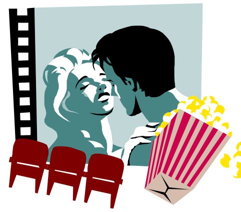
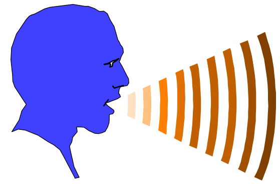
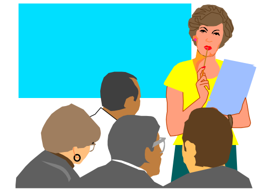
   

EE3414
Multimedia Communication Systems - I

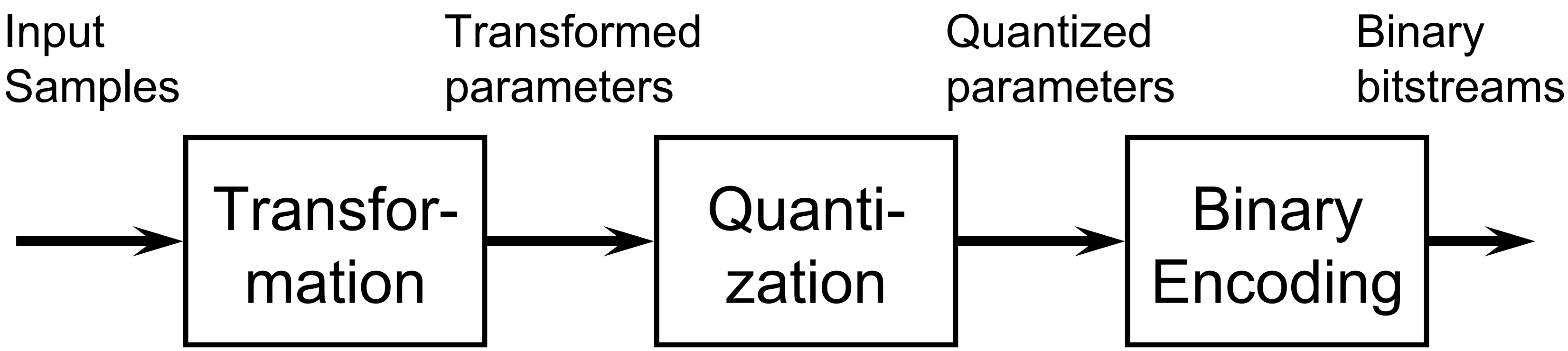
DCT and Transform Coding

Yao Wang
Polytechnic University, Brooklyn, NY11201
http://eeweb.poly.edu

Outline

- Transform coding
  - General principle
- DCT
  - Definition, basis images
  - Energy distribution
  - Approximation with different number of basis images
  - Quantization of DCT coefficients

A Typical Compression System



Prediction
Transforms
Model fitting
……

Scalar Q
Vector Q

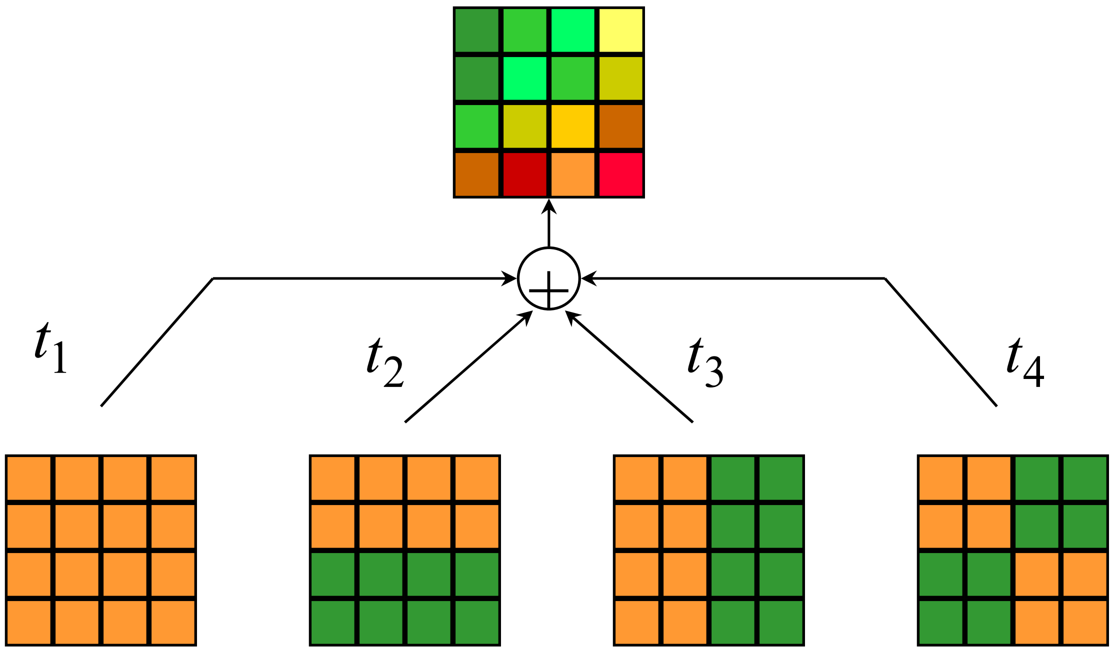
Fixed length
Variable length (Huffman, arithmetic, LZW)

Motivation for “Transformation”

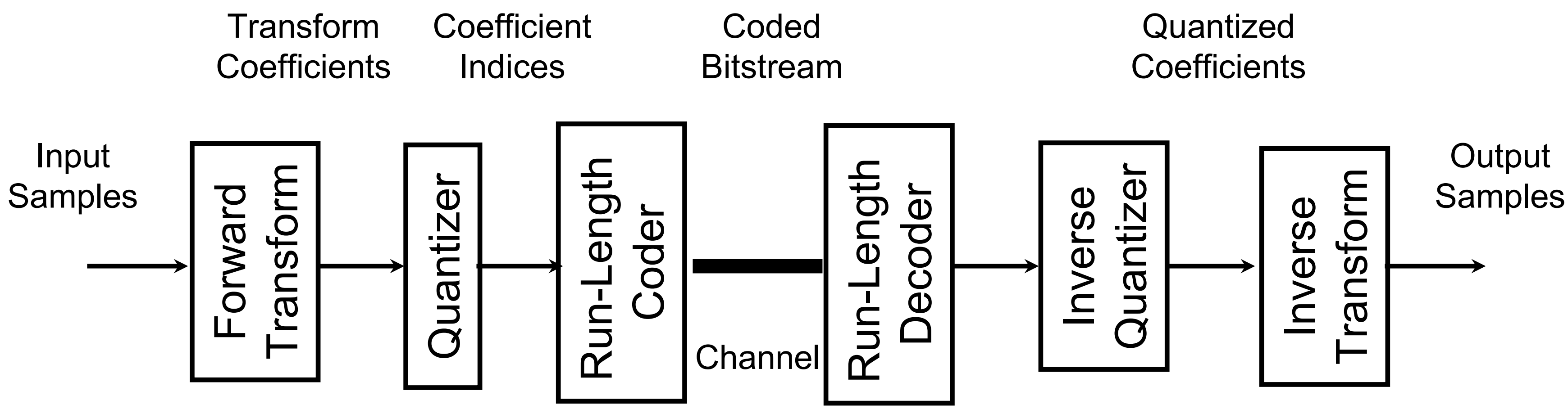
- Motivation for transformation:
  - To yield a more efficient representation of the original samples.
  - The transformed parameters should require fewer bits to code.
- Types of transformation used:
  - For speech coding: prediction
    - Code predictor and prediction error samples
  - For audio coding: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

Transform Coding

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



A Typical Transform Coder



Transform Basis Design

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

1D Unitary Transform

Consider the $N-$point signal $s(n)$ as an N - dimensional vector

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

The inverse transform says that $\mathbf{s}$ can be represented as the sum of N basis vectors

$$\mathbf{s} = t_0\mathbf{u_0} + t_1\mathbf{u}_1 + \ldots + t_{N-1}\mathbf{u}_{N-1}$$

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

$$\mathbf{u}_k = \begin{bmatrix} u_{k,0} \\ u_{k,1} \\ \ldots \\ u_{k,N-1} \end{bmatrix}$$

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

$$t_k = (\mathbf{u}_k, \mathbf{s}) = \sum_{n=0}^{N-1} u_{k;n}^{*} s_n$$

1D Discrete Cosine Transform

- Can be considered “real” version of DFT
  - Basis vectors contain only co-sinusoidal patterns

DFT

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

$$\mathbf{u}_k = \frac{1}{\sqrt{N}}\begin{bmatrix} \cos\left(\frac{2\pi k}{N}0\right) \\ \cos\left(\frac{2\pi k}{N}1\right) \\ \dots \\ \cos\left(\frac{2\pi k}{N}(N-1)\right) \end{bmatrix} + j\frac{1}{\sqrt{N}}\begin{bmatrix} \sin\left(\frac{2\pi k}{N}0\right) \\ \sin\left(\frac{2\pi k}{N}1\right) \\ \dots \\ \sin\left(\frac{2\pi k}{N}(N-1)\right) \end{bmatrix}$$

DCT

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

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

$$\mathbf{u}_k = \alpha(k)\begin{bmatrix} \cos\left(\frac{\pi k}{2N}1\right) \\ \cos\left(\frac{\pi k}{2N}3\right) \\ \dots \\ \cos\left(\frac{\pi k}{2N}(2N+1)\right) \end{bmatrix}$$

Example: 4-point DCT

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

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

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

Go through on the board

2D Separable Transform

Consider the $M \times N -$ point image $\mathbf{S}$ as a $M \times N$ - dimensional array (matrix)

$$\mathbf{S} = \begin{bmatrix} S_{0,0} & S_{0,1} & \dots & S_{0,N-1} \\ S_{1,0} & S_{1,1} & \dots & S_{1,N-1} \\ \dots & \dots & \dots & \dots \\ S_{M-1,0} & S_{M-1,1} & \dots & S_{M-1,N-1} \end{bmatrix}$$

The inverse transform says that $\mathbf{s}$ can be represented as the sum of $M \times N$ basis images

$$\mathbf{S} = T_{0,0}\mathbf{U}_{\mathbf{0},0} + T_{0,1}\mathbf{U}_{0,1} + \dots + T_{M-1,N-1}\mathbf{U}_{M-1,N-1}$$

where $\mathbf{U}_{k,l}$ corresponds to the (k,l) - th transform kernel :

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

The forward transform says that the expansion coefficient S_k can be determined by the inner product of $\mathbf{S}$ and $\mathbf{U}_{k,l}$:

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

2D Discrete Cosine Transform

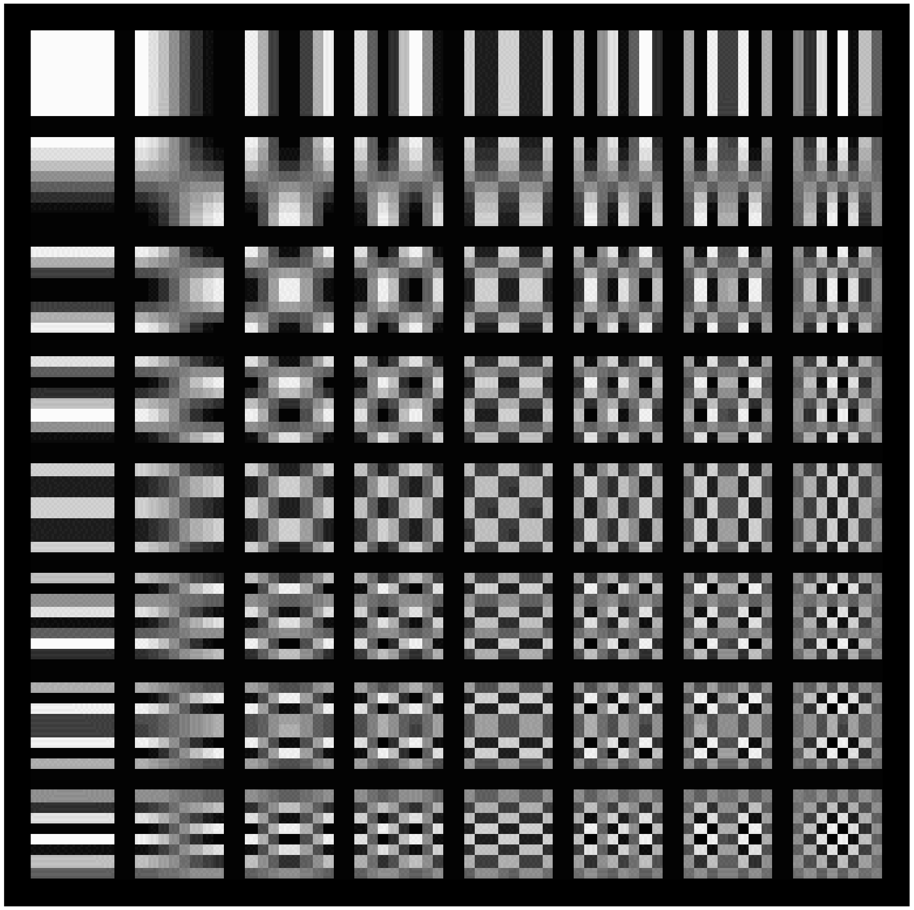
- Basis image = outer product of 1D DCT basis vector

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

$$\mathbf{U}_{k,l;M,N} = \mathbf{u}_{k;M}\left(\mathbf{u}_{l;N}\right)^T$$

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

Basis Images of 8x8 DCT



Example: 4x4 DCT

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

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

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

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

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

Using $T_{k,l}=(\mathbf{U}_{k,l},\mathbf{S})$ yields, e.g,

$$T_{0,0}=(\mathbf{U}_{0,0},\mathbf{S})=\frac{1}{4}\left(\begin{bmatrix}1 & 1 & 1 & 1\\ 1 & 1 & 1 & 1\\ 1 & 1 & 1 & 1\\ 1 & 1 & 1 & 1\end{bmatrix},\begin{bmatrix}1 & 2 & 2 & 0\\ 0 & 1 & 3 & 1\\ 0 & 1 & 2 & 1\\ 1 & 2 & 2 & -1\end{bmatrix}\right)=\frac{1}{4}\left(1+2+2+0+0+1+3+1+0+1+2+1+1+2+2-1\right)=\frac{18}{4}=4.5$$

$$T_{2,2}=(\mathbf{U}_{3,2},\mathbf{S})=\frac{1}{4}\left(\begin{bmatrix}1 & -1 & -1 & 1\\ -1 & 1 & 1 & -1\\ -1 & 1 & 1 & -1\\ 1 & -1 & -1 & 1\end{bmatrix},\begin{bmatrix}1 & 2 & 2 & 0\\ 0 & 1 & 3 & 1\\ 0 & 1 & 2 & 1\\ 1 & 2 & 2 & -1\end{bmatrix}\right)=\frac{1}{4}\left(1-2-2+1+3-1+1+2-1+1-2-2-1\right)=-\frac{2}{4}=-0.5$$

Completing calculation using Matlab yields (using "dct2(S)"):

$$\mathbf{T}=\begin{bmatrix}4.5 & -0.0793 & -3 & 1.1152\\ 0.4619 & -0.5 & 0.1913 & 0\\ 0 & 2.0391 & -0.5 & -0.3034\\ -0.1913 & 0 & 0.4619 & -0.5\end{bmatrix}$$

Reconstruction from top 2x2 coefficients only yields:

$$\mathrm{S}=\begin{bmatrix}1.09 & 1.54 & 1.38 & 0.86\\ 1.44 & 1.9 & 1.63 & 0.95\\ 1.11 & 1.38 & 1.63 & 0.95\\ 0.61 & 0.68 & 0.31 & 0.05\end{bmatrix}$$

The reconstructed image varies slower than the original, because we cut off some high frequency coefficients

DCT on a Real Image Block

```
>>imblock = lena256(128:135,128:135)
imblock=
182 196 199 201 203 201 199 173
175 180 176 142 148 152 148 120
148 118 123 115 114 107 108 107
115 110 110 112 105 109 101 100
104 106 106 102 104 95 98 105
99 115 131 104 118 86 87 133
112 154 154 107 140 97 88 151
145 158 178 123 132 140 138 133
```

```
>>dctblock =dct2(imblock)
dctblock=1.0e+003*
1.0550 0.0517 0.0012 -0.0246 -0.0120 -0.0258 0.0120 0.0232
0.1136 0.0070 -0.0139 0.0432 -0.0061 0.0356 -0.0134 –0.0130
0.1956 0.0101 -0.0087 -0.0029 -0.0290 -0.0079 0.0009 0.0096
0.0359 -0.0243 -0.0156 -0.0208 0.0116 -0.0191 -0.0085 0.0005
0.0407 -0.0206 -0.0137 0.0171 -0.0143 0.0224 -0.0049 -0.0114
0.0072 -0.0136 -0.0076 -0.0119 0.0183 -0.0163 -0.0014 –0.0035
-0.0015 -0.0133 -0.0009 0.0013 0.0104 0.0161 0.0044 0.0011
-0.0068 -0.0028 0.0041 0.0011 0.0106 -0.0027 -0.0032 0.0016
```

Low frequency coefficients (top left corner) are much larger than the rest!

Reconstructed Block

Original block

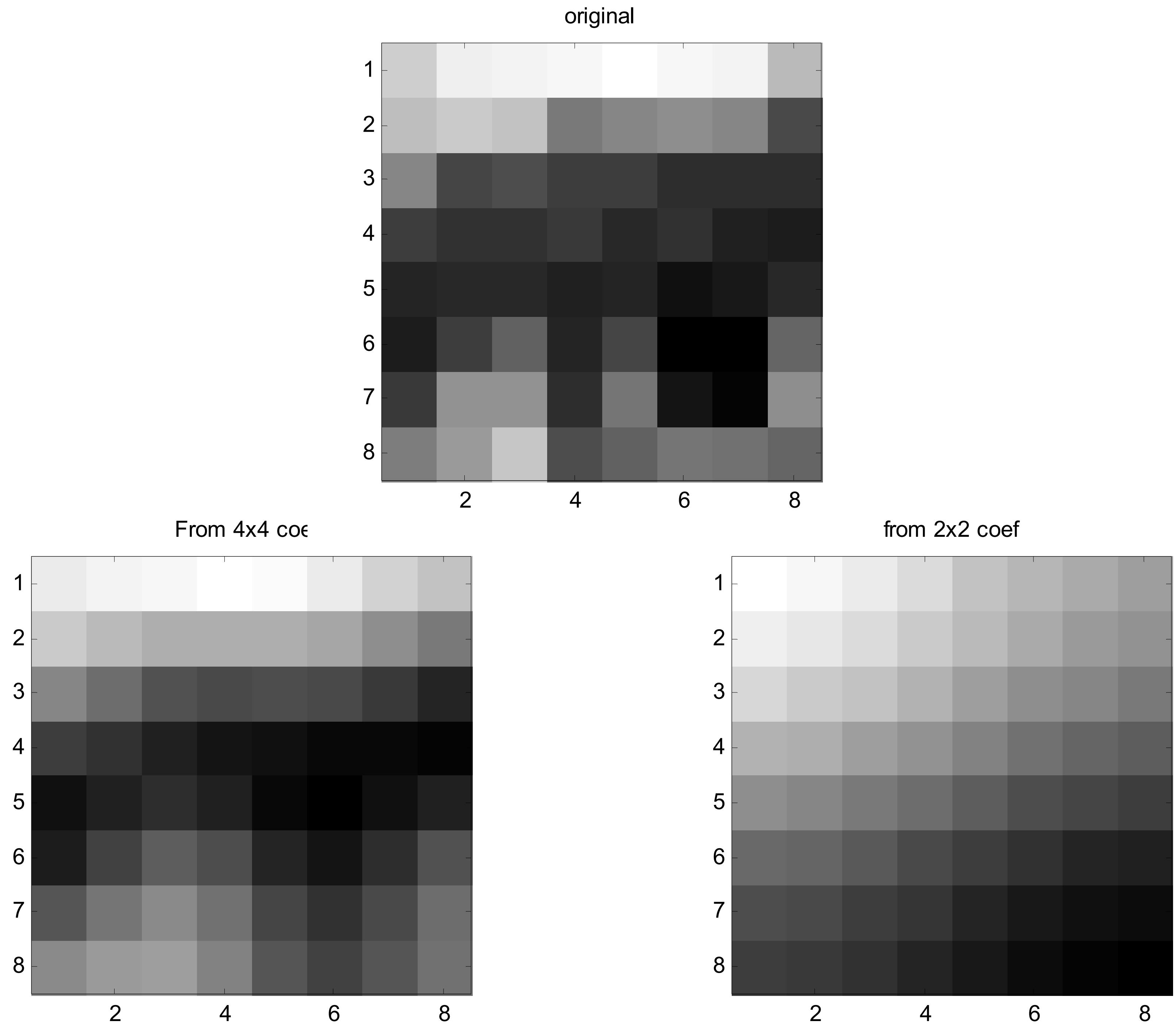
182	196	199	201	203	201	199	173
175	180	176	142	148	152	148	120
148	118	123	115	114	107	108	107
115	110	110	112	105	109	101	100
104	106	106	102	104	95	98	105
99	115	131	104	118	86	87	133
112	154	154	107	140	97	88	151
145	158	178	123	132	140	138	133

Reconstructed using top 4x4 coefficients

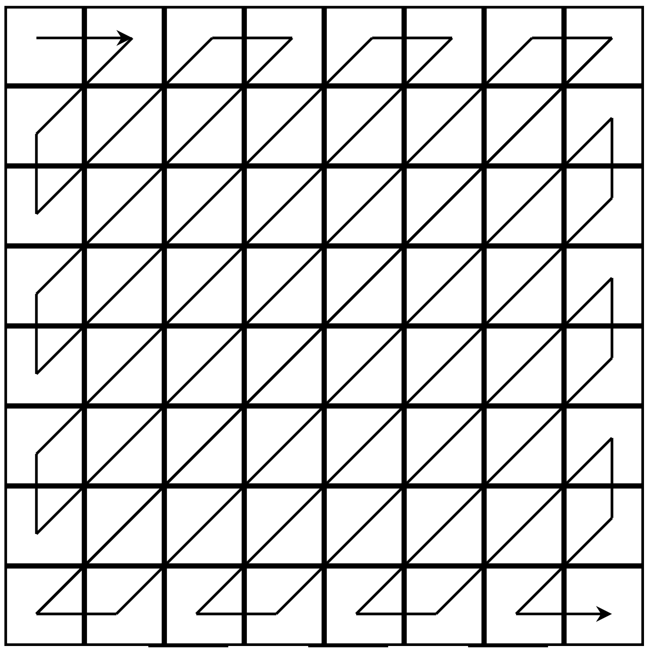
190	192	195	197	196	189	179	172
175	169	163	163	164	160	150	141
147	136	124	120	122	121	114	106
115	110	103	98	96	94	93	92
96	104	109	104	93	89	96	104
102	117	130	123	106	98	109	124
126	139	148	138	119	111	121	136
148	155	156	144	125	118	126	138

Reconstructed using top 2x2 coefficients only

162	161	158	154	149	146	143	141
159	157	154	151	147	143	140	138
153	151	149	145	141	137	135	133
145	144	141	138	134	131	128	126
137	135	133	130	126	123	121	119
129	128	125	122	119	116	114	113
123	122	119	117	114	111	109	108
119	118	116	114	111	108	106	105


original
From 4x4 coe
from 2x2 coef
1
2
3
4
5
6
7
8
2
4
6
8

Zig-Zag Ordering of DCT Coefficients



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

DCT Coefficient Variance Distribution

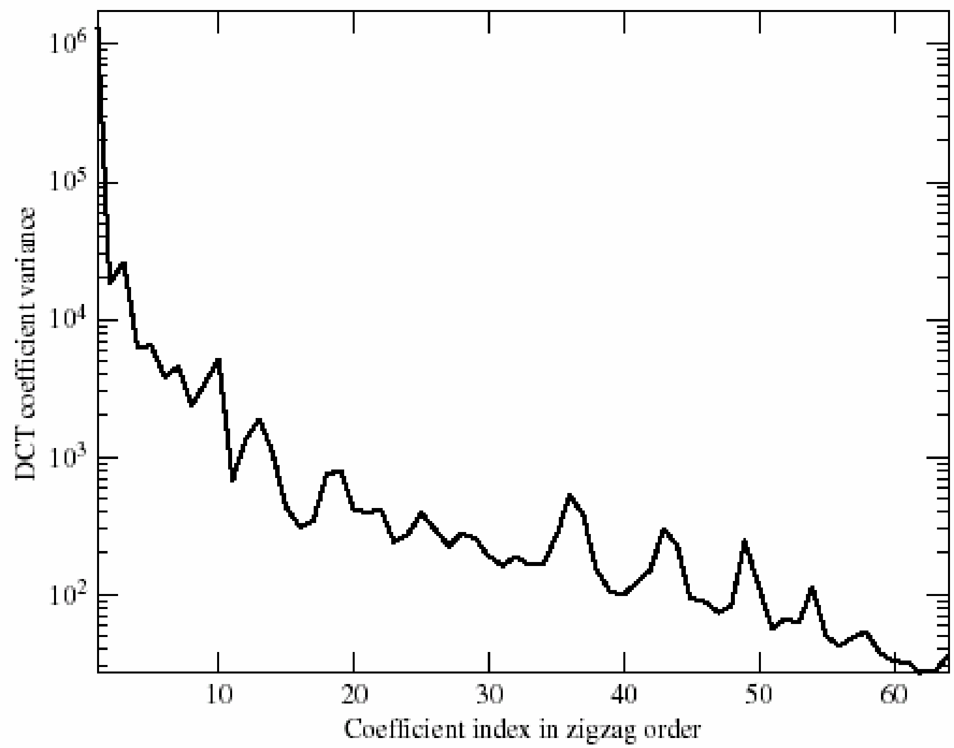
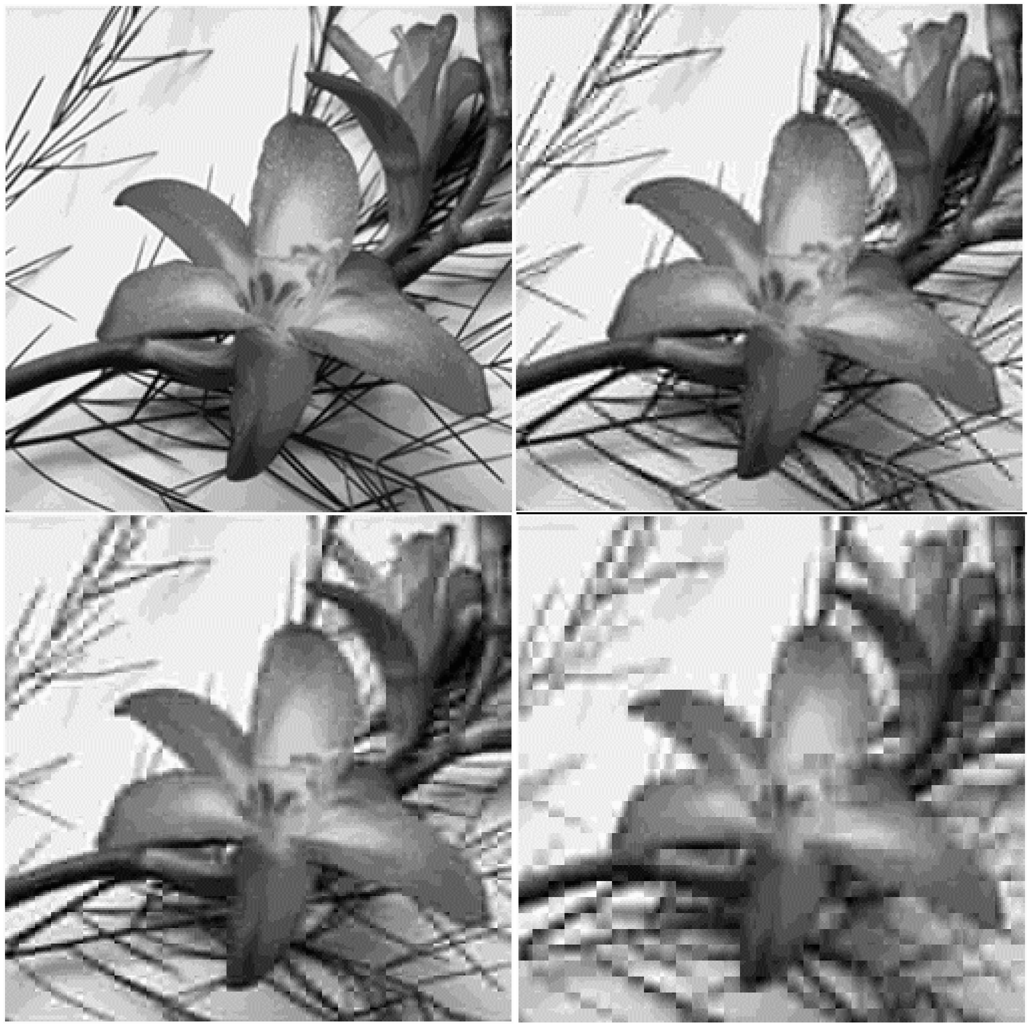


Figure 9.4 Energy distribution of the 8 × 8 DCT coefficients of the test image "flower."

Low frequency coefficients have much higher variances.

Variance for 1 coefficient is the mean of the coefficient square over all 8x8 blocks

Approximation by DCT Basis



Original

With 16/64 Coefficients

With 8/64 Coefficients

With 4/64 Coefficients

Matlab demo: dctdemo

VCdemo

- Can show DCT images as collection or as blocks
- Can perform quantization, show quantized image and bit rate

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
  - Normalization matrix can then be scaled by a scale factor (QP)

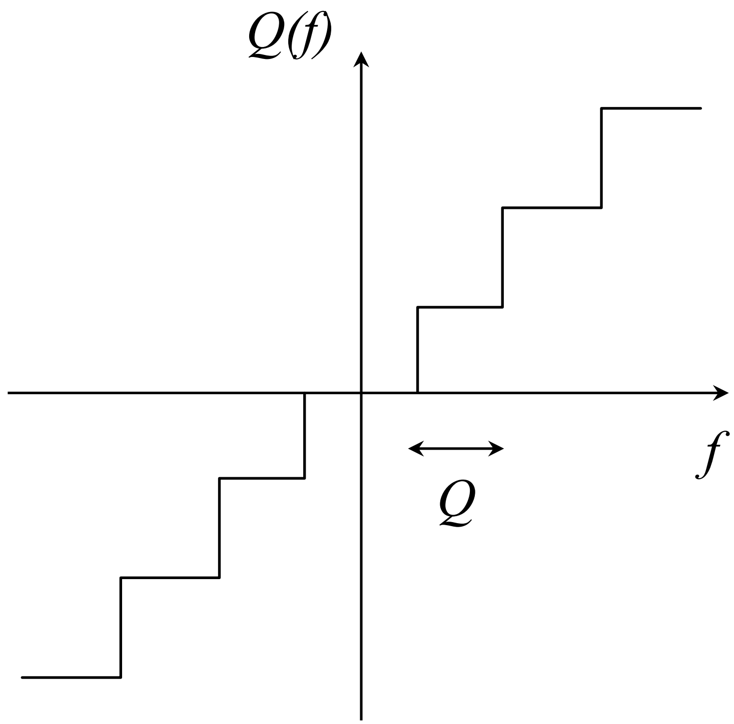
Default Normalization Matrix in JPEG

For Luminance component

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

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

Uniform Quantization



Qindex(f)=floor((f+Q/2)/Q)
Q(f)=Qindex(f)*Q

Example: Quantized Indices

```
>>dctblock =dct2(imblock)
dctblock=1.0e+003*
   1.0550    0.0517    0.0012   -0.0246   -0.0120   -0.0258    0.0120    0.0232
   0.1136    0.0070   -0.0139    0.0432   -0.0061    0.0356   -0.0134   –0.0130
   0.1956    0.0101   -0.0087   -0.0029   -0.0290   -0.0079    0.0009    0.0096
   0.0359   -0.0243   -0.0156   -0.0208    0.0116   -0.0191   -0.0085    0.0005
   0.0407   -0.0206   -0.0137    0.0171   -0.0143    0.0224   -0.0049   -0.0114
   0.0072   -0.0136   -0.0076   -0.0119    0.0183   -0.0163   -0.0014   –0.0035
  -0.0015   -0.0133   -0.0009    0.0013    0.0104    0.0161    0.0044    0.0011
  -0.0068   -0.0028    0.0041    0.0011    0.0106   -0.0027   -0.0032    0.0016
```

```
>>QP=1;
>>QM=Qmatrix*QP;
>>qdct=floor((dctblock+QM/2)./(QM))
qdct =
   66    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=

1056	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=1.0e+003*

1.0550	0.0517	0.0012	-0.0246	-0.0120	-0.0258	0.0120	0.0232
0.1136	0.0070	-0.0139	0.0432	-0.0061	0.0356	-0.0134	–0.0130
0.1956	0.0101	-0.0087	-0.0029	-0.0290	-0.0079	0.0009	0.0096
0.0359	-0.0243	-0.0156	-0.0208	0.0116	-0.0191	-0.0085	0.0005
0.0407	-0.0206	-0.0137	0.0171	-0.0143	0.0224	-0.0049	-0.0114
0.0072	-0.0136	-0.0076	-0.0119	0.0183	-0.0163	-0.0014	–0.0035
-0.0015	-0.0133	-0.0009	0.0013	0.0104	0.0161	0.0044	0.0011
-0.0068	-0.0028	0.0041	0.0011	0.0106	-0.0027	-0.0032	0.0016

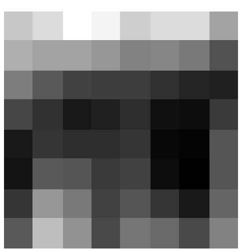
Example: Reconstructed Image

%reconstructed image block

```
>>  qimblock=round(idct2(iqdct))
```

qimblock=

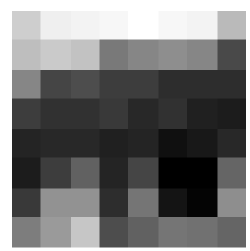
186	196	213	207	189	196	195	166
173	166	167	161	150	152	147	126
149	130	117	116	115	109	104	101
120	109	97	102	108	93	91	113
97	111	107	108	112	89	87	128
95	131	127	114	117	91	84	129
112	160	146	118	129	112	98	136
131	182	158	122	144	139	121	151



Original image block

imblock=

182	196	199	201	203	201	199	173
175	180	176	142	148	152	148	120
148	118	123	115	114	107	108	107
115	110	110	112	105	109	101	100
104	106	106	102	104	95	98	105
99	115	131	104	118	86	87	133
112	154	154	107	140	97	88	151
145	158	178	123	132	140	138	133



Matlab Implementation

```
% This function does the following processing on each block: DCT, quantize DCT,
    inverse DCT
% This function is called by dct_exp
% Yao Wang 4/10/2003

function qimblock=blkdct_quant(imblock,QP)
Qmatrix=[16,11,10,16,26,40,51,61;
   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];

dctblock=dct2(imblock);
QM=Qmatrix*QP;
qdct=floor((dctblock+QM/2)./QM);
iqdct=qdct.*QM;
qimblock=round(idct2(iqdct));
```

Matlab Implementation

```
%For each image block, perform DCT, quantization and inverse DCT
%User can specify the QP for quantization
%Yao Wang, 4/10/2003
%This program calls function "blkdct_quant"

function dct_exp(fname,QP)
img=imread(fname);
qimg=blkproc(img,[8 8],'blkdct_quant',QP);
figure;
imagesc(qimg),axis image, truesize, axis off; colormap(gray);
title('DCT Domain Quantized Image');
```

Note: the “blkproc” function in matlab performs “blkdct_quant” on each 8x8 block of the image. Using this function is much faster than using a loop that going through each block.

lena256_gray.tif

QP=1

QP=0.5

QP=2

What Should You Know

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

References

Gonzalez and Woods, *Digital image processing*, 2nd edition, Prentice Hall, 2002. Sec.8.5.2.

Vcdemo site:

http://www-ict.its.tudelft.nl/~inald/vcdemo/

This site also contain pdf slides of lectures on transform coding.