IMAGE COMPRESSION

- Data redundancy
- Self-information and Entropy
- Error-free and lossy compression
- Huffman coding
- Predictive coding
- Transform coding
Data Redundancy

- **CODING**: Fewer bits to represent frequent symbols.
- **INTERPIXEL / INTERFRAME**: Neighboring pixels have similar values.
- **PSYCHOVISUAL**: Human visual system can not simultaneously distinguish all colors.

Coding Redundancy (contd.)

- Consider equation (A): It makes sense to assign fewer bits to those \( r_k \) for which \( p_r(r_k) \) are large in order to reduce the sum.
- this achieves data compression and results in a variable length code.
- More probable gray levels will have fewer # of bits.

\[
L_{\text{avg}} = \sum_{k=0}^{L-1} l(r_k) \cdot p_r(r_k) \quad \rightarrow (A)
\]
8.2.9: Predictive coding

To reduce / eliminate interpixel redundancies

**Lossless predictive coding: ENCODER**

\[
e_n = f_n - \hat{f}_n
\]
### Decoder

- **Compressed image** → **symbol decoder** → **prediction error** → **original image**

#### Prediction error:
- $e_n = f_n - \hat{f}_n$
- $e_n$ is coded using a variable length code (symbol encoder)
- $f_n = e_n + \hat{f}_n$

### Example

**Example 1:**
- $\hat{f}_n = \text{Int} \left( \sum_{i=1}^{m} \alpha_i f_{n-i} \right)$
- $\Rightarrow$ Linear predictor; $m =$ order of predictor

**Example 2:**
- $\hat{f}_n (x,y) = \text{Int} \left( \sum_{i=0}^{m} \alpha_i f(x,y-i) \right)$

In 2-D:
- **Past information**
- **Current pixel** $f(x,y)$
- $f(x,y) = \sum \alpha (x', y') f(x', y)$
- $(x', y') \in W_{xy}$
Predictive coding and temporal redundancy

\[ f(x, y, t) = \text{round}[\alpha f(x, y, t - 1)] \]

\[ e(x, y, t) = \hat{f}(x, y, t) - f(x, y, t) \]

**FIGURE 8.35**
(a) and (b) Two views of Earth from an orbiting space shuttle video. (c) The prediction error image resulting from Eq. (8.2-36). (d) A histogram of the prediction error. (Original images courtesy of NASA.)

An example of first order predictor

Histograms of original and error images.
Lossy Compression (pp. 596)

Lossy compression: uses a quantizer to compress further the number of bits required to encode the 'error'.

First consider this:

\[
\Sigma \quad e \quad \hat{e}_n \quad \text{Enc} \quad \hat{e}_n \quad \text{Dec} \quad \Sigma
\]

\[
f_n \quad \hat{f}_n \quad e \neq \hat{e}_n \Rightarrow \hat{f}_n \neq \tilde{f}_n
\]

Notice that, unlike in the case of loss-less prediction, in lossy prediction the predictors P "see" different inputs at the encoder and decoder.

Quantization error

This results in a gradual buildup of error which is due to the quantization error at the encoder site.

In order to minimize this buildup of error due to quantization we should ensure that 'Ps' have the same input in both the cases.

<table>
<thead>
<tr>
<th>n</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
</tr>
</thead>
<tbody>
<tr>
<td>( f_n )</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>13</td>
<td>14</td>
</tr>
<tr>
<td>( \hat{f}_n )</td>
<td>( f_{n-1} )</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( e_n )</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>( \hat{e}_n )</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>. . .</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>( \hat{f}_n )</td>
<td>0</td>
<td>2</td>
<td>4</td>
<td>6</td>
<td>8</td>
<td>10</td>
<td>. . .</td>
<td>6</td>
<td>8</td>
<td>10</td>
<td>. . .</td>
<td>6</td>
<td>8</td>
<td>10</td>
<td>. . .</td>
</tr>
</tbody>
</table>
Predictive Coding With Feedback

\[ f(x,y) \]

\[ e_n \]

\[ \hat{f}_n = e_n + \hat{f}_n \]

This feedback loop prevents error building.

\[ f_n = e_n + \hat{f}_n \]

Compressed image

Symbol decoder

\[ f_n = e_n + \hat{f}_n \]

uncompressed image

Example

Example:

\[ \hat{f}_n = \alpha \hat{f}_{n-1} \]

and \[ \hat{e}_n = \begin{cases} +\xi & e_n > 0 \\ -\xi & e_n < 0 \end{cases} \]

\[ \hat{f}_n = \hat{e}_n + \hat{f}_n \]

\[ = \hat{e}_n + \alpha \hat{f}_{n-1} \]

\[ 0 < \alpha < 1 \]

prediction coefficient
Example

with feedback

\[
\begin{aligned}
    f_n &= 0 \quad 1 \quad 2 \quad 3 \quad 4 \\
    e_n &= 1 \quad 2 \quad 1 \quad 2 \\
    \hat{e}_n &= 0 \quad 2 \quad 0 \quad 2 \\
    \hat{f}_n &= 0 \quad 0 \quad 2 \quad 2 \\
    \hat{\hat{f}}_n &= 0 \quad 0 \quad 2 \quad 2 \\
\end{aligned}
\]

Note: The quantizer used here is \( \text{floor}(e_n/2) \times 2 \). This is different from the one used in the earlier example. Note that this would result in a worse response if used without Feedback (output will be flat at \( "0" \)).

Another example

\{14, 15, 14, 15, 13, 15, 14, 20, 26, 27, 28, 27, 29, 37, 37, 62, 75, 77, 78, 79, 80, 81, 81, 82, 82\}

Note: The quantizer used here is \( \text{floor}(e_n/2) \times 2 \). This is different from the one used in the earlier example. Note that this would result in a worse response if used without Feedback (output will be flat at \( "0" \)).
A comparison (Fig 8.23)

**FIGURE 8.23** A 512 × 512 8-bit monochrome image.

Four linear predictors

**FIGURE 8.24** A comparison of four linear prediction techniques.
Motion compensation in Video

Block Transform Coding

Section 8.2.8
Transform coding

- **Construct n x n subimages**
- **Forward transform**
- **Quantizer**
- **Symbol encoder**
  - Compressed image

Compressed image

- **Symbol decoder**
- **Inverse transform**
- **Merge n x n subimages**
  - Uncompressed image

**Blocking artifact:** boundaries between subimages become visible

Transform Selection

- DFT
- Discrete Cosine Transform (DCT)
- Wavelet transform
- Karhunen-Loeve Transform (KLT)
- …

\[
T(u,v) = \sum_{x=0}^{n-1} \sum_{y=0}^{n-1} g(x,y)r(x,y,u,v) \tag{8.2.10}
\]

\[
g(x,y) = \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} T(u,v)s(x,y,u,v) \tag{8.2.11}
\]
Transform Kernels

- Separable if \( r(x, y, u, v) = r_1(x, u)r_2(y, v) \) (8.2.12)
- E.g. DFT: \( r(x, y, u, v) = \exp(-j2\pi(ux + vy)/n) \)
- E.g. Walsh-Hadamard transform (see page 568, text)
- E.g. DCT

Approximations using DFT, Hadamard and DCT, and the scaled error images
Discrete Cosine Transform


1-D Case: Extended 2N Point Sequence

Consider 1-D first; Let $x(n)$ be a $N$ point sequence $0 \leq n \leq N - 1$. Let $y(n)$ be a $2N$ point sequence $0 \leq n \leq 2N - 1$. Define $Y(u)$ as $2N$ point DFT of $y(n)$.

$$y(n) = x(n) + x(2N - 1 - n) = \begin{cases} x(n), & 0 \leq n \leq N - 1 \\ x(2N - 1 - n), & N \leq n \leq 2N - 1 \end{cases}$$
The N-point DCT of \( x(n) \), \( C(u) \), is given by

\[
C(u) = \exp \left( -j \frac{2\pi}{2N} \cdot un \right)
\]

\[
= \sum_{n=0}^{N-1} x(n) \exp \left( -j \frac{2\pi}{2N} \cdot un \right) + \sum_{n=N}^{2N-1} x(2N - 1 - n) \exp \left( -j \frac{2\pi}{2N} \cdot un \right)
\]

\[
= \sum_{n=0}^{N-1} x(n) \exp \left( -j \frac{2\pi}{2N} \cdot un \right) + \sum_{m=0}^{N-1} x(m) \exp \left( -j \frac{2\pi}{2N} \cdot u(2N - 1 - m) \right)
\]

\[
= \exp \left( j \frac{\pi}{2N} u \right) \sum_{n=0}^{N-1} x(n) \exp \left( -j \frac{\pi}{2N} u \right) - j \frac{2\pi}{2N} \cdot un
\]

\[
+ \exp \left( j \frac{\pi}{2N} u \right) \sum_{n=0}^{N-1} x(n) \exp \left( j \frac{\pi}{2N} u + j \frac{2\pi}{2N} \cdot un \right)
\]

\[
= \exp \left( j \frac{\pi}{2N} u \right) \sum_{n=0}^{N-1} x(n) \cos \left( \frac{\pi}{2N} u(2n + 1) \right)
\]

---

The N-point DCT of \( x(n) \), \( C(u) \), is given by

\[
C(u) = \begin{cases} 
\exp \left( -j \frac{\pi}{2N} u \right) Y(u), & 0 \leq u \leq N-1 \\
0 & \text{otherwise.}
\end{cases}
\]

The unitary DCT transformations are:

\[
F(u) = \alpha(u) \sum_{n=0}^{N-1} f(n) \cos \left( \frac{\pi}{2N} (2n + 1)u \right), \quad 0 \leq u \leq N - 1, \quad \text{where}
\]

\[
\alpha(0) = \frac{1}{\sqrt{N}}, \quad \alpha(u) = \sqrt{\frac{2}{N}} \quad \text{for} \quad 1 \leq k \leq N - 1.
\]

The inverse transformation is

\[
f(n) = \sum_{u=0}^{N-1} \alpha(u) F(u) \cos \left( \frac{\pi}{2N} (2n + 1)u \right), \quad 0 \leq u \leq N - 1.
\]
Discrete Cosine Transform—In 2-D

\[ C(u, v) = \alpha(u)\alpha(v) \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f(x, y) \cos \left( \frac{(2x+1)u\pi}{2N} \right) \cos \left( \frac{(2y+1)v\pi}{2N} \right) \]

for \( u, v = 0, 1, 2, \ldots, N - 1 \), where

\[ \alpha(u) = \begin{cases} \sqrt{\frac{1}{N}} & \text{for } u = 0 \\ \sqrt{\frac{2}{N}} & \text{for } u = 1, 2, \ldots, N-1. \end{cases} \]

\[ f(x, y) = \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} \alpha(u)\alpha(v) C(u, v) \cos \left( \frac{(2x+1)u\pi}{2N} \right) \cos \left( \frac{(2y+1)v\pi}{2N} \right) \]

for \( x, y = 0, 1, 2, \ldots, N - 1 \)

---

DCT Summary

\[ T(u, v) = \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f(x, y) r(x, y, u, v) \]

\[ r(x, y, u, v) = s(x, y, u, v) \]

\[ = \alpha(u)\alpha(v) \cos \left( \frac{(2x+1)u\pi}{2N} \right) \cos \left( \frac{(2y+1)v\pi}{2N} \right) \]
DCT Basis functions

**FIGURE 8.30** Discrete-cosine basis functions for \( N = 4 \). The origin of each block is at its top left.

Implicit Periodicity-DFT vs DCT (Fig 8.32)

**FIGURE 8.32** The periodicity implicit in the 1-D (a) DFT and (b) DCT.
Why DCT?

- Blocking artifacts less pronounced in DCT than in DFT.
- Good approximation to the Karhunen-Loeve Transform (KLT) but with basis vectors fixed.
- DCT is used in JPEG image compression standard.

Sub-image size selection (fig 8.26)

**Figure 8.33**
Reconstruction error versus subimage size.
Different sub-image sizes

![Image of sub-images]

**Figure 8.34** Approximations of Fig. 8.23 using 25% of the DCT coefficients: (a) and (b) 8 x 8 subimage results; (c) zoomed original; (d) 2 x 2 result; (e) 4 x 4 result; and (f) 8 x 8 result.

Bit Allocation/Threshold Coding

- # of coefficients to keep
- How to quantize them
  - Threshold coding
  - Zonal coding

**Threshold coding**

For each subimage i
- Arrange the transform coefficients in decreasing order of magnitude
- Keep only the top X% of the coefficients and discard rest.
- Code the retained coefficient using variable length code.
Zonal Coding

Zonal coding:

1. Compute the variance of each of the transform coeff; use the subimages to compute this.
2. Keep X% of their coeff. which have maximum variance.
3. Variable length coding (proportional to variance)

Bit allocation: In general, let the number of bits allocated be made proportional to the variance of the coefficients. Suppose the total number of bits per block is B. Let the number of retained coefficients be M. Let \( v(i) \) be variance of the \( i \)-th coefficient. Then

\[
b(i) = \frac{B}{M} + \frac{1}{2} \log_2 v(i) - \frac{1}{2} \sum_{i=1}^{M} \log_2 v(i)
\]
Typical Masks (Fig 8.36)

FIGURE 8.36 A
(a) typical zonal mask, (b) zonal bit allocation, (c) threshold mask, and (d) thresholded coefficient ordering sequence. Shading highlights the coefficients that are retained.

Image Approximations

FIGURE 8.35 Approximations of Fig. 8.25 using 12.5% of the 8 × 8 DCT coefficients (a), (c), and (e) thresholding results, (b), (d), and (f) zonal ordering results.
The JPEG standard

- The following is mostly from Tekalp’s book, Digital Video Processing by M. Tekalp (Prentice Hall).
- For the new JPEG-2000 check out the web site www.jpeg.org.
- See Example 8.17 in the text

JPEG (contd.)

- JPEG is a lossy compression standard using DCT.
- Four modes of operation: Sequential (baseline), hierarchical, progressive, and lossless.
- Arbitrary image sizes; DCT mode 8-12 bits/sample. Luminance and chrominance channels are separately encoded.
- We will only discuss the baseline method.
JPEG-baseline.

- **DCT**: The image is divided into 8x8 blocks. Each pixel is level shifted by $2^{n-1}$ where $2^n$ is the maximum number of gray levels in the image. Thus for 8 bit images, you subtract 128. Then the 2-D DCT of each block is computed. For the baseline system, the input and output data precision is restricted to 8 bits and the DCT values are restricted to 11 bits.
- **Quantization**: the DCT coefficients are threshold coded using a quantization matrix, and then reordered using zig-zag scanning to form a 1-D sequence.
- The non-zero AC coefficients are Huffman coded. The DC coefficients of each block are DPCM coded relative to the DC coefficient of the previous block.

JPEG -color image

- **RGB to Y-Cr-Cb space**
  - $Y = 0.3R + 0.6G + 0.1B$
  - $Cr = \frac{1}{2}(B-Y) + 0.5$
  - $Cb = \frac{1}{1.6}(R-Y) + 0.5$
- Chrominance samples are sub-sampled by 2 in both directions.

<table>
<thead>
<tr>
<th>Y1</th>
<th>Y2</th>
<th>Y3</th>
<th>Y4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Y5</td>
<td>Y6</td>
<td>Y7</td>
<td>Y8</td>
</tr>
<tr>
<td>Y9</td>
<td>Y10</td>
<td>Y11</td>
<td>Y12</td>
</tr>
<tr>
<td>Y13</td>
<td>Y14</td>
<td>Y15</td>
<td>Y16</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Cr1</th>
<th>Cr2</th>
<th>Cr3</th>
<th>Cr4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cr1</td>
<td>Cr2</td>
<td>Cr3</td>
<td>Cr4</td>
</tr>
<tr>
<td>Cb1</td>
<td>Cb2</td>
<td>Cb3</td>
<td>Cb4</td>
</tr>
</tbody>
</table>

**Non-Interleaved**
Scan 1: Y1, Y2, ..., Y16
Scan 2: Cr1, Cr2, Cr3, Cr4
Scan 3: Cb1, Cb2, Cb3, Cb4

**Interleaved**: Y1, Y2, Y3, Y4, Cr1, Cb1, Y5, Y6, Y7, Y8, Cr2, Cb2, ...
JPEG – quantization matrices

- Check out the matlab workspace (dctex.mat).
- Quantization table for the luminance channel.
- Quantization table for the chrominance channel.
- JPEG baseline method
  - Consider the 8x8 image (matlab: array s.)
  - Level shifted (s-128=sd).
  - 2d-DCT: dct2(sd)= dcts
  - After dividing by quantization matrix qmat: dctshat.
  - Zigzag scan as in threshold coding.

   \[20, 5, -3, -1, -2, -3, 1, 1, -1, -1, 0, 0, 1, 2, 3, -2, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, EOB\].

An 8x8 sub-image (s)

\[s = (8x8\text{block})\]

\[
\begin{array}{cccccccccccccc}
183 & 160 & 94 & 153 & 194 & 163 & 132 & 165 \\
183 & 153 & 116 & 176 & 167 & 166 & 130 & 169 \\
179 & 168 & 171 & 182 & 179 & 170 & 131 & 167 \\
177 & 177 & 179 & 177 & 165 & 165 & 131 & 167 \\
178 & 178 & 179 & 176 & 182 & 164 & 130 & 171 \\
179 & 180 & 180 & 179 & 183 & 164 & 130 & 171 \\
179 & 179 & 180 & 182 & 183 & 170 & 129 & 173 \\
180 & 179 & 181 & 179 & 181 & 170 & 130 & 169 \\
\end{array}
\]

\[sd = (\text{level shifted})\]

\[
\begin{array}{cccccccccccccc}
55 & 32 & -34 & 25 & 66 & 35 & 4 & 37 \\
55 & 25 & -12 & 48 & 59 & 38 & 2 & 41 \\
51 & 40 & 43 & 54 & 51 & 42 & 3 & 39 \\
49 & 49 & 51 & 49 & 51 & 37 & 3 & 39 \\
50 & 50 & 51 & 48 & 54 & 36 & 2 & 43 \\
51 & 52 & 52 & 51 & 55 & 36 & 2 & 43 \\
51 & 51 & 52 & 54 & 55 & 42 & 1 & 45 \\
52 & 51 & 53 & 51 & 53 & 42 & 2 & 41 \\
\end{array}
\]
2D DCT (dcts) and the quantization matrix (qmat)

\[ \text{dcts} = \begin{bmatrix}
312 & 56 & -27 & 17 & 79 & -60 & 26 & -26 \\
-38 & -28 & 13 & 45 & 31 & -1 & -24 & -10 \\
-20 & -18 & 10 & 33 & 21 & -6 & -16 & -9 \\
-11 & -7 & 9 & 15 & 10 & -11 & -13 & 1 \\
-6 & 1 & 6 & 5 & -4 & -7 & -5 & 5 \\
3 & 3 & 0 & -2 & -7 & -4 & 1 & 2 \\
3 & 5 & 0 & -4 & -8 & -1 & 2 & 4 \\
3 & 1 & -1 & -2 & -3 & -1 & 4 & 1
\end{bmatrix} \]

\[ \text{qmat} = \begin{bmatrix}
16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\
12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\
14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\
14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\
18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\
24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\
49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\
72 & 92 & 95 & 98 & 112 & 100 & 103 & 99
\end{bmatrix} \]

Division by qmat (dcthat)=dcts/qmat

\[ \text{dcthat} = \begin{bmatrix}
20 & 5 & -3 & 1 & 3 & -2 & 1 & 0 \\
-3 & -2 & 1 & 2 & 1 & 0 & 0 & 0 \\
-1 & -1 & 1 & 1 & 1 & 0 & 0 & 0 \\
-1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix} \]

\[ \text{dcts} = \begin{bmatrix}
312 & 56 & -27 & 17 & 79 & -60 & 26 & -26 \\
-38 & -28 & 13 & 45 & 31 & -1 & -24 & -10 \\
-20 & -18 & 10 & 33 & 21 & -6 & -16 & -9 \\
-11 & -7 & 9 & 15 & 10 & -11 & -13 & 1 \\
-6 & 1 & 6 & 5 & -4 & -7 & -5 & 5 \\
3 & 3 & 0 & -2 & -7 & -4 & 1 & 2 \\
3 & 5 & 0 & -4 & -8 & -1 & 2 & 4 \\
3 & 1 & -1 & -2 & -3 & -1 & 4 & 1
\end{bmatrix} \]
### Zig-zag scan of dcthat

<table>
<thead>
<tr>
<th>dcthat=</th>
<th>Zigzag scan as in threshold coding.</th>
</tr>
</thead>
<tbody>
<tr>
<td>20  5  -3  1  3</td>
<td>[20, 5, -3, -1, -2, -3, 1, 1, -1,</td>
</tr>
<tr>
<td>-3  -2  1  2  1</td>
<td>-1, 0, 0, 1, 2, 3, -2, 1, 1, 0,</td>
</tr>
<tr>
<td>-1  -1  1  1  1</td>
<td>0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,</td>
</tr>
<tr>
<td>-1  0  0  1  0</td>
<td>0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,</td>
</tr>
<tr>
<td>0  0  0  0  0  0</td>
<td>0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0</td>
</tr>
<tr>
<td>0  0  0  0  0  0</td>
<td>0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0</td>
</tr>
<tr>
<td>0  0  0  0  0  0</td>
<td>0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0</td>
</tr>
</tbody>
</table>

### Threshold coding -revisited

**Zig-zag scanning of the coefficients.**

The coefficients along the zig-zag scan lines are mapped into [run,level] where the level is the value of non-zero coefficient, and run is the number of zero coeff. preceding it. The DC coefficients are usually coded separately from the rest.
JPEG – baseline method example

Zigzag scan as in threshold coding.

[20, 5, -3, -1, -2, -3, 1, 1, -1, -1, 0, 0, 1, 2, 3, -2, 1, 1, 0, 0, 0, 0, 1, 1, 0, 1, EOB].

- The DC coefficient is DPCM coded (difference between the DC coefficient of the previous block and the current block.)
- The AC coeff. are mapped to run-level pairs.
  
  (0,5), (0,-3), (0,-1), (0,-2), (0,-3), (0,1), (0,1), (0,-1), (0,-1), (2,1), (0,2), (0,3), (0,-2), (0,1), (0,1), (6,1), (0,1), (1,1), EOB.
- These are then Huffman coded (codes are specified in the JPEG scheme.)
- The decoder follows an inverse sequence of operations. The received coefficients are first multiplied by the same quantization matrix.
  
  \( \text{Recddcthat} = \text{dcthat} \times \text{qmat} \).
- Compute the inverse 2-D dct. \( \text{Recdsd} = \text{idct2}(\text{Recddcthat}) \); add 128 back.
  
  \( \text{Recds} = \text{Recdsd} + 128 \).

Decoder

\[
\begin{array}{ccccccccccccccccccc}
320 & 55 & -30 & 16 & 72 & -80 & 51 & 0 & 67 & 12 & -9 & 20 & 69 & 43 & -8 & 42 \\
-36 & -24 & 14 & 38 & 26 & 0 & 0 & 0 & 58 & 25 & 15 & 30 & 65 & 40 & -4 & 47 \\
-14 & -13 & 16 & 24 & 40 & 0 & 0 & 0 & 46 & 41 & 44 & 40 & 59 & 38 & 0 & 49 \\
-14 & 0 & 0 & 29 & 0 & 0 & 0 & 0 & 41 & 52 & 59 & 43 & 57 & 42 & 3 & 42 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 44 & 54 & 58 & 40 & 58 & 47 & 3 & 33 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 49 & 52 & 53 & 40 & 61 & 47 & 1 & 33 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 53 & 50 & 53 & 46 & 63 & 41 & 0 & 45 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 55 & 50 & 56 & 53 & 64 & 34 & -1 & 57
\end{array}
\]
Received signal

<table>
<thead>
<tr>
<th>Reconstructed S=</th>
<th>$S = (8\times8\text{block})$</th>
</tr>
</thead>
<tbody>
<tr>
<td>195 140 119 148 197 171 120 170</td>
<td>183 160 94 153 194 153 132 165</td>
</tr>
<tr>
<td>186 153 143 158 193 168 124 175</td>
<td>183 153 116 176 187 166 130 169</td>
</tr>
<tr>
<td>174 169 172 168 187 166 128 177</td>
<td>179 168 171 182 179 170 131 167</td>
</tr>
<tr>
<td>169 180 187 171 185 170 131 170</td>
<td>177 177 179 177 179 165 131 167</td>
</tr>
<tr>
<td>172 182 186 168 186 175 131 161</td>
<td>178 178 179 176 182 164 130 171</td>
</tr>
<tr>
<td>177 180 181 168 189 175 129 161</td>
<td>179 180 180 179 183 164 130 171</td>
</tr>
<tr>
<td>181 178 181 174 191 169 128 173</td>
<td>179 179 180 182 183 170 129 173</td>
</tr>
<tr>
<td>183 178 184 181 192 162 127 185</td>
<td>180 179 181 179 181 170 130 169</td>
</tr>
</tbody>
</table>

Example

FIGURE 6.28: Left column: Approximations of Fig. 6.27 using the DCT and normalization array of Fig. 6.27(b). Right column: Similar results for A2.
Wavelet Compression

**FIGURE 8.39** A wavelet coding system:
(a) encoder; (b) decoder.

**FIGURE 8.41** (a), (c), and (e) Wavelet coding results with a compression ratio of 108 to 1: (b), (d), and (f) similar results for a compression of 167 to 1.
Image Compression: Summary

- Data redundancy
- Self-information and Entropy
- Error-free compression
  - Huffman coding, Arithmetic coding, LZW coding, Run-length encoding
  - Predictive coding
- Lossy coding techniques
  - Predictive coding (Lossy)
  - Transform coding
    - DCT, DFT, KLT, ...
- JPEG image compression standard