Data Redundancy

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

Coding Redundancy (contd.)

- Consider equation (A): It makes sense to assign fewer bits to those \( r_k \) for which \( p(r_k) \) 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 \( n \) of bits.

General Model

- **General compression model**
  - Source encoder
  - Channel encoder
  - Channel decoder
  - Source decoder

Predictive coding

- **To reduce / eliminate interpixel redundancies**
- **Lossless predictive coding**: ENCODER
  - Lossless predictive coding: ENCODER
  - Prediction error
  - \( e_n = f_n - \hat{f}_n \)
Decoder

Compressed image

Symbol decoder

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

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

Prediction error:

Compressed image

Predictor

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

\[ \Sigma \]

Original image

Example

Example 1: \( \hat{f} = \text{Int} \)

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

\[ \Sigma \]

\[ \alpha \]

\[ (x', y') \in W_{xy} \]

\[ f(x, y) \]

\[ x \]

\[ y \]

Example 2: \( \hat{f}(x, y) = \text{Int} \left( \sum_{i} \alpha (x', y') f(x', y) \right) \)

\[ (x', y') \in W_{xy} \]

\[ f(x, y) \]

\[ x \]

\[ y \]

Lossy Compression (Section 8.5)

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

First consider this:

\[ \Sigma \]

\[ e \]

\[ Q \]

\[ \text{Enc} \]

\[ f(x, y) \]

\[ \hat{f}(x, y) \]

\[ \hat{e}(x, y) \]

\[ \Sigma \]

\[ f_n \]

\[ e_n \]

\[ \hat{f}_n \]

\[ \hat{e}_n \]

\[ \hat{f}_n \]

\[ \hat{e}_n \]

\[ P \]

\[ e \neq \hat{e} \Rightarrow \hat{f} \neq \hat{f} \]

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.

\[ f_n \]

\[ 0 \ 1 \ 2 \ 3 \ 4 \ 5 \ 6 \ 7 \ 8 \ 9 \ 10 \ 11 \ 12 \ 13 \ 14 \]

\[ \hat{f}_n \]

\[ f_n = \hat{f}_n - 1 \]

\[ e_n \]

\[ 0 \ 1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 1 \]

\[ \hat{e}_n \]

\[ 0 \ 2 \ 2 \ 2 \ 2 \ . . . \ . . . \]

\[ \hat{f}_n \]

\[ 0 \ 2 \ 4 \ 6 \ 8 \ 10 \ . . . \ . . . \]

Predictive Coding With Feedback

\[ f(x, y) \]

\[ e_n \]

\[ \hat{f}_n \]

\[ \hat{e}_n \]

\[ \Sigma \]

\[ \text{symbol encoder} \]

\[ \text{Compressed image} \]

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

\[ \hat{e}_n \]

\[ \hat{f}_n \]

\[ \hat{e}_n \]

\[ \hat{f}_n \]

\[ \text{Compressed image} \]

\[ \text{uncompressed image} \]

This feedback loop prevents error building.
Example

\[ f_n = \alpha f_{n-1} \]

and \( e_n = \begin{cases} +\epsilon & e_n > 0 \\ -\epsilon & e_n < 0 \end{cases} \)

\[ \hat{f}_n = e_n + \alpha f_{n-1} \]

Example: with feedback

\[ f_n = \begin{array}{cccc} 0 & 1 & 2 & 3 \\ \epsilon & 1 & 2 & 1 \end{array} \]

\[ e_n = \begin{array}{cccc} 0 & 2 & 0 & 2 \end{array} \]

\[ f_{n-1} = \begin{array}{cccc} 0 & 0 & 2 & 2 \end{array} \]

\[ f_{n+1} = \begin{array}{cccc} 0 & 0 & 2 & 2 \end{array} \]

Note: The quantizer used here is \( \text{floor}(e_n/2)\cdot 2 \). 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, 20, 27, 29, 27, 29, 27, 27, 31, 37, 37, 75, 77, 78, 79, 80, 81, 82, 82)\}

A comparison (Fig 8.23)

Four linear predictors

Transform Coding
Transform coding

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

Transform Selection

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

Discrete Cosine Transform


1-D Case: Extended 2N Point Sequence

Consider 1-D first; let \( x(n) \) be a \( N \) point sequence \( 0 \leq n \leq N - 1 \).

\[
\begin{align*}
\text{Forward transform:} & \quad Y(u) = \sum_{n=0}^{N-1} x(n) e^{-j2\pi nu/N}, \\
\text{Inverse transform:} & \quad x(n) = \sum_{u=0}^{N-1} Y(u) e^{j2\pi nu/N}.
\end{align*}
\]

**DCT & DFT**

\[
F(u) = \sum_{n=0}^{N-1} y(n) e^{-j2\pi nu/N} = \sum_{n=0}^{N-1} x(n) e^{-j2\pi nu/N}.
\]
DCT

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

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

The unitary DCT transformations are:

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

where

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

The inverse transformation is

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

---

Discrete Cosine Transform—in 2-D

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

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

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

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

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

---

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.

---

Karhunen-Loeve Transform

- READ pp. 476 and Section 11.4 Text (if you are working on the face recognition project, you must!)
- Also called the Hotelling Transform.
- Transform is data dependent. Let \( X \) denote the random data, and let \( C \) be the covariance matrix of \( X \), i.e.

\[
C = E(X - m)(X - m)^T
\]

where \( m \) is the mean vector of the data.

The matrix \( C \) is real and symmetric, and hence can be diagonalized using its eigenvectors.
What are eigenvectors?

Let \( C = \text{E}[ (x - m)(x - m)^T] \)

where \( C \) is the \( N \times N \) covariance matrix, \( x \) is an \( N \)-dimensional vector, and \( m \) is the mean vector of the samples. The eigenvectors \( \phi \) of \( C \) are given by

\[
\phi_i = \lambda_i, \quad \text{where } \lambda_i \text{ are the corresponding eigenvalues.}
\]

Now consider a matrix \( A \) whose columns correspond to the eigenvectors of \( C \), arranged such that the first column corresponds to the eigenvector with the largest eigenvalue, and second with the second largest and so on. Then, consider a transformation on the vectors \( x \) such that

\[
y = A^T(x - m),
\]

Note that \( y \) is zero mean, and its covariance matrix is

\[
C_y = \text{E}[yy^T] = \text{E}[A^T(x - m)(x - m)^T A^T] = A^T C A = \Lambda
\]

where \( \Lambda \) is diagonal with eigenvalues, in decreasing order. Note that the elements of the transformed vector are uncorrelated (non-diagonal elements are zero).

What is KLT

The transformation from \( X \) to \( Y \) is called the KLT. In computing the transformation matrix \( A \), we assume that the columns are made up of orthonormal eigenvectors (i.e., the inverse of \( A \) is also its transpose.) Thus the basis vectors of the KLT are the orthonormal eigenvectors of the covariance matrix \( C \).

The KLT yields uncorrelated coefficients.

For compression, we only keep the top \( K \) coefficients corresponding to the \( K \) largest eigenvectors.

Then the mean squared error in reconstruction is given by

\[
MSE = \sum_{j=K+1}^{N} \lambda_j
\]

Sub-image size selection

Different sub-image sizes

Bit Allocation/Threshold Coding

- Number 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

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

\[
h(i) = \frac{1}{B} + \frac{1}{2}\log_2 v(i) - \frac{1}{M} \log_2 \sum_{i} v(i)
\]
Zonal Mask & bit allocation: an example

Typical Masks (Fig 8.36)

Image Approximations

The JPEG standard

JPEG (contd.)

JPEG-baseline.
JPEG - color image

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

Chrominance samples:
- \( Y16, Y15, Y14, Y13 \)
- \( Y12, Y11, Y10, Y9 \)
- \( Y8, Y7, Y6, Y5 \)
- \( Y4, Y3, Y2, Y1 \)
- \( Cr4, Cr3, Cr2, Cr1 \)
- \( Cb4, Cb3, Cb2, Cb1 \)

Non-interleaved scan:
Scan 1: \( Y1, Y2, \ldots, Y16 \)
Scan 2: \( Cr1, Cr2, Cr3, Cr4 \)
Scan 3: \( Cb1, Cb2, Cb3, Cb4 \)

Interleaved scan:
\( Y1, Y2, Y3, Y4, Cr1, Cb1, Y5, Y6, Y7, Y8, Cr2, Cb2, \ldots \)

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`: `dcthat`.
  - Zigzag scan as in threshold coding:
    - `[20, 5, -3, -1, -2, -3, 1, 1, -1, -1, 0, 0, 0, 1, 2, 3, -2, -1, 1, 0, 0, 0, 0, 0, 0]`

An 8x8 sub-image (s)

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

<table>
<thead>
<tr>
<th></th>
<th>183</th>
<th>160</th>
<th>94</th>
<th>153</th>
<th>194</th>
<th>163</th>
<th>132</th>
<th>165</th>
</tr>
</thead>
<tbody>
<tr>
<td>163</td>
<td>153</td>
<td>116</td>
<td>178</td>
<td>187</td>
<td>166</td>
<td>130</td>
<td>169</td>
<td></td>
</tr>
<tr>
<td>179</td>
<td>168</td>
<td>171</td>
<td>192</td>
<td>179</td>
<td>170</td>
<td>131</td>
<td>167</td>
<td></td>
</tr>
<tr>
<td>177</td>
<td>177</td>
<td>179</td>
<td>179</td>
<td>165</td>
<td>161</td>
<td>167</td>
<td></td>
<td></td>
</tr>
<tr>
<td>178</td>
<td>178</td>
<td>179</td>
<td>182</td>
<td>164</td>
<td>130</td>
<td>171</td>
<td></td>
<td></td>
</tr>
<tr>
<td>179</td>
<td>180</td>
<td>180</td>
<td>173</td>
<td>184</td>
<td>130</td>
<td>171</td>
<td></td>
<td></td>
</tr>
<tr>
<td>179</td>
<td>180</td>
<td>180</td>
<td>183</td>
<td>184</td>
<td>130</td>
<td>173</td>
<td></td>
<td></td>
</tr>
<tr>
<td>180</td>
<td>179</td>
<td>181</td>
<td>181</td>
<td>181</td>
<td>180</td>
<td>170</td>
<td>169</td>
<td></td>
</tr>
</tbody>
</table>

sd (level shifted)

<table>
<thead>
<tr>
<th>35</th>
<th>32</th>
<th>-34</th>
<th>25</th>
<th>65</th>
<th>35</th>
<th>4</th>
<th>37</th>
</tr>
</thead>
<tbody>
<tr>
<td>55</td>
<td>25</td>
<td>-12</td>
<td>48</td>
<td>59</td>
<td>38</td>
<td>2</td>
<td>41</td>
</tr>
<tr>
<td>51</td>
<td>40</td>
<td>43</td>
<td>54</td>
<td>51</td>
<td>43</td>
<td>3</td>
<td>39</td>
</tr>
<tr>
<td>49</td>
<td>49</td>
<td>51</td>
<td>48</td>
<td>54</td>
<td>36</td>
<td>2</td>
<td>43</td>
</tr>
<tr>
<td>51</td>
<td>52</td>
<td>52</td>
<td>51</td>
<td>55</td>
<td>36</td>
<td>2</td>
<td>43</td>
</tr>
<tr>
<td>51</td>
<td>52</td>
<td>52</td>
<td>54</td>
<td>55</td>
<td>43</td>
<td>1</td>
<td>43</td>
</tr>
<tr>
<td>52</td>
<td>53</td>
<td>51</td>
<td>53</td>
<td>42</td>
<td>2</td>
<td>41</td>
<td></td>
</tr>
</tbody>
</table>

2D DCT (dcts) and the quantization matrix (qmat)

\( \text{dcts} = \)

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

\( \text{qmat} = \)

\[\begin{bmatrix}
16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\
12 & 14 & 19 & 26 & 58 & 60 & 55 & 3 \\
14 & 13 & 24 & 40 & 57 & 89 & 56 & -11 \\
14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\
18 & 22 & 37 & 65 & 68 & 109 & 103 & -6 \\
24 & 35 & 55 & 64 & 81 & 104 & 113 & 3 \\
49 & 64 & 78 & 87 & 103 & 121 & 101 & 3 \\
72 & 95 & 98 & 112 & 103 & 99 & 1 \\
\end{bmatrix}\]

Division by qmat (dcthat) = dcts/qmat

\( \text{dcthat} = \)

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

Zig-zag scan of dcthat

\( \text{dcthat} = \)

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

Zigzag scan as in threshold coding:
\[\{20, 5, -3, -1, -2, -3, 1, 1, -1, -1, 0, 0, 1, 2, 3, -2, -1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 1, \text{EOB}\} \]
Threshold coding - revisited

Zig-zag scanning of the coefficients.

The coefficients along the zig-zag scan lines are mapped into \([\text{run, level}]\) where

- \(\text{level}\) is the value of non-zero coefficient, and
- \(\text{run}\) 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:

\([20, 5, -3, -1, -2, 3, 1, -1, -1, 0, 1, 2, 3, -2, 1, 1, 0, 0, 0, 0, 1, 1, \text{EOB}]\).

- The DC coefficient is DPCM coded (difference between the DC coefficient of
  the previous block and the current block.)
- The AC coefficients are mapped to run-level pairs, \((0,5), (0, -1), (0, -2), (0, -3), (0, 1), (0, -1), (0, -1), (0, -2), (0, 1), (0, -1), (0, -1), \text{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. \((\text{recdsd} = \text{idct2}(\text{recddcthat}); \text{add 128 back.})\)
  \(\text{recd} = \text{recdsd} + 128\).

Decoder

\[\begin{array}{cccccccc}
320 & 55 & -30 & 16 & 72 & -80 & 51 & 0 \\
-36 & -24 & 14 & 38 & 26 & 0 & 0 & 0 \\
-14 & -13 & 16 & 24 & 40 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{array}\]

\[\text{Recddcthat} = \text{dcthat} \times \text{qmat}\]

Recddcthat =

\[\begin{array}{cccccccc}
67 & 12 & 0 & 20 & 69 & 43 & 8 & 42 \\
58 & 25 & 15 & 30 & 65 & 40 & 4 & 47 \\
46 & 41 & 44 & 40 & 59 & 38 & 0 & 49 \\
41 & 52 & 59 & 43 & 57 & 42 & 3 & 42 \\
44 & 54 & 68 & 40 & 58 & 47 & 3 & 33 \\
49 & 52 & 53 & 40 & 61 & 47 & 1 & 33 \\
53 & 50 & 63 & 46 & 63 & 41 & 0 & 46 \\
55 & 50 & 56 & 53 & 64 & 34 & -1 & 57 \\
\end{array}\]

Received signal

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

Reconstructed S=

\[\begin{array}{cccccccc}
155 & 140 & 119 & 108 & 97 & 87 & 77 & 67 \\
186 & 153 & 129 & 105 & 86 & 76 & 66 & 56 \\
174 & 147 & 122 & 102 & 83 & 73 & 63 & 53 \\
169 & 138 & 109 & 89 & 79 & 69 & 59 & 49 \\
172 & 145 & 116 & 87 & 77 & 67 & 57 & 47 \\
177 & 150 & 120 & 100 & 80 & 70 & 60 & 50 \\
183 & 156 & 126 & 106 & 86 & 76 & 66 & 56 \\
181 & 154 & 124 & 104 & 84 & 74 & 64 & 54 \\
\end{array}\]

Example

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