Lossy Image Compression Methods

- Scalar quantization (SQ).
- Vector quantization (VQ).
- DCT Compression
  - JPEG
- Wavelet Compression
  - SPIHT
  - UWIC (University of Washington Image Coder)
  - EBCOT (JPEG 2000)

JPEG Standard

- JPEG - Joint Photographic Experts Group
- JPEG 2000 uses wavelet compression.

Barbara

- Original
- JPEG
- VQ
- Wavelet-SPIHT

32:1 compression ratio
0.25 bits/pixel (8 bits)
Images and the Eye

- Images are meant to be viewed by the human eye (usually).
- The eye is very good at "interpolation", that is, the eye can tolerate some distortion. So lossy compression is not necessarily bad. The eye has more acuity for luminance (gray scale) than chrominance (color).
  - Gray scale is more important than color.
  - Compression is usually done in the YUV color coordinates, Y for luminance and U, V for color.
  - U and V should be compressed more than Y
  - This is why we will concentrate on compressing gray scale (8 bits per pixel) images.

Distortion

- Lossy compression: \( x \neq \hat{x} \)
- Measure of distortion is commonly mean squared error (MSE). Assume \( x \) has \( n \) real components (pixels).
  \[
  MSE = \frac{1}{n} \sum_{i=1}^{n} (x_i - \hat{x}_i)^2
  \]

PSNR

- Peak Signal to Noise Ratio (PSNR) is the standard way to measure fidelity.
  \[
  PSNR = 10 \log_{10} \left( \frac{m^2}{MSE} \right)
  \]
  where \( m \) is the maximum value of a pixel possible. For gray scale images (8 bits per pixel) \( m = 255 \).
- PSNR is measured in decibels (dB).
  - .5 to 1 dB is said to be a perceptible difference.
  - Decent images start at about 30 dB

Rate-Fidelity Curve

- Properties:
  - Increasing
  - Slope decreasing
**PSNR is not Everything**

PSNR = 25.8 dB  
PSNR = 25.8 dB

**Lossy Compression Example**

- Gray scale image, 8 bits per pixel
- Codebook size 16, 4 bits or less per pixel
- Compression is $8/4 = 2:1$ or better with entropy coding of indices.
- We’ll see later that it is better to do “Vector Quantization” where the codebook has 2x2 or 4x4 blocks.

**Scalar Quantization**

```
<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

```
<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

Decoded Image

```
<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

```
<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

**PSNR Reflects Fidelity (1)**

PSNR 25.8

.63 bpp

12.8 : 1

**PSNR Reflects Fidelity (2)**

PSNR 24.2

.31 bpp

25.6 : 1

**PSNR Reflects Fidelity (3)**

PSNR 23.2

.16 bpp

51.2 : 1
Scalar Quantization Strategies

- Build a codebook with a training set. Encode and decode with fixed codebook.
  - Most common use of quantization
- Build a codebook for each image. Transmit the codebook with the image.
- Training can be slow.

Distortion

- Let the image be pixels \( x_1, x_2, \ldots, x_T \).
- Define \( \text{index}(x) \) to be the index transmitted on input \( x \).
- Define \( c(j) \) to be the codeword indexed by \( j \).
  \[
  D = \sum_{i=1}^{T} (x_i - c(\text{index}(x_i)))^2 \quad \text{(Distortion)}
  
  \text{MSE} = \frac{D}{T}
  \]

Uniform Quantization Example

- 512 x 512 image with 8 bits per pixel.
- 8 codewords

<table>
<thead>
<tr>
<th>Codebook</th>
<th>Index</th>
<th>Codeword</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>47</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>79</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>111</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>143</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>175</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>207</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>239</td>
</tr>
</tbody>
</table>

Encoder

<table>
<thead>
<tr>
<th>input code</th>
<th>000</th>
<th>001</th>
<th>010</th>
<th>011</th>
<th>100</th>
<th>101</th>
<th>110</th>
<th>111</th>
</tr>
</thead>
<tbody>
<tr>
<td>boundary</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>codeword</td>
<td>239</td>
<td>207</td>
<td>239</td>
<td>207</td>
<td>111</td>
<td>143</td>
<td>175</td>
<td>143</td>
</tr>
</tbody>
</table>

Decoder

<table>
<thead>
<tr>
<th>output code</th>
<th>000</th>
<th>001</th>
<th>010</th>
<th>011</th>
<th>100</th>
<th>101</th>
<th>110</th>
<th>111</th>
</tr>
</thead>
<tbody>
<tr>
<td>boundary</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>codeword</td>
<td>16</td>
<td>47</td>
<td>79</td>
<td>111</td>
<td>143</td>
<td>175</td>
<td>207</td>
<td>239</td>
</tr>
</tbody>
</table>

Bit rate = 3 bits per pixel
Compression ratio = 8/3 = 2.67

Improve Bit Rate

Frequency of pixel values

\( q_j \) = probability that a pixel is coded to index \( j \)
Potential average bit rate is entropy,

\[
H = \sum_j q_j \log_2 \left( \frac{1}{q_j} \right)
\]

Example

- 512 x 512 image = 216,144 pixels

<table>
<thead>
<tr>
<th>index</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>
</tr>
</thead>
<tbody>
<tr>
<td>frequency</td>
<td>35,000</td>
<td>30,635</td>
<td>44,954</td>
<td>96,127</td>
<td>108,154</td>
<td>160,191</td>
<td>192,220</td>
<td>224,255</td>
</tr>
</tbody>
</table>

Huffman Tree

ABR = (100000 x 1 + 90000 x 2 + 43000 x 4 + 39144 x 5)/216144 = 2.997
Arithmetic coding should work better.
**Improving Distortion**

- Choose the codeword as a weighted average

Let $p_x$ be the probability that a pixel has value $x$. Let $[L_j, R_j]$ be the input interval for index $j$. $c(j)$ is the codeword indexed $j$.

$$c(j) = \text{round} \left( \sum_{x \in [L_j, R_j]} x p_x \right)$$

---

**Example**

All pixels have the same index.

<table>
<thead>
<tr>
<th>pixel value</th>
<th>frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>100</td>
</tr>
<tr>
<td>9</td>
<td>100</td>
</tr>
<tr>
<td>10</td>
<td>40</td>
</tr>
<tr>
<td>11</td>
<td>20</td>
</tr>
<tr>
<td>12</td>
<td>20</td>
</tr>
<tr>
<td>13</td>
<td>10</td>
</tr>
<tr>
<td>14</td>
<td>9</td>
</tr>
<tr>
<td>15</td>
<td>8</td>
</tr>
</tbody>
</table>

New Codeword: round

Old Distortion = 10000

New Distortion = 200

---

**An Extreme Case**

Frequency of pixel values

Only two codewords are ever used!!

---

**Non-uniform Scalar Quantization**

Frequency of pixel values

- codeword
- boundary between codewords

---

**Lloyd Algorithm**

- Lloyd (1957)
- Creates an optimized codebook of size $n$.
- Let $p_x$ be the probability of pixel value $x$.
- Probabilities might come from a training set.
- Given codewords $c(0), c(1), \ldots, c(n-1)$ and pixel $x$ let $\text{index}(x)$ be the index of the closest code word to $x$.
- Expected distortion is

$$D = \sum_x p_x (x - \text{index}(x))^2$$

- Goal of the Lloyd algorithm is to find the codewords that minimize distortion.
- Lloyd finds a local minimum by an iteration process.

---

**Lloyd Algorithm**

Choose a small error tolerance $\varepsilon > 0$.
Choose start codewords $c(0), c(1), \ldots, c(n-1)$
Compute $X(j) = \{ x : x \text{ is a pixel value closest to } c(j) \}$
Compute distortion $D$ for $c(0), c(1), \ldots, c(n-1)$
Repeat
- Compute new codewords

$$c'(j) = \text{round} \left( \sum_{x \in X(j)} x p_x \right)$$

- Compute $X'(j) = \{ x : x \text{ is a pixel value closest to } c'(j) \}$
- Compute distortion $D'$ for $c(0), c(1), \ldots, c(n-1)$
- if $|D - D'|/D < \varepsilon$ then quit
- else $c := c' ; X := X' ; D := D'$
End(repeat)
Example
Initially c(0) = 2 and c(1) = 5

\[
\begin{array}{cccccccc}
\text{pixel value} & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\
\text{frequency} & 100 & 100 & 100 & 40 & 30 & 20 & 10 & 0 \\
\end{array}
\]

\[
X(0) = \{0.3\}, X(1) = \{4.7\} \\
D(0) = 140 \ F^1 + 100 \ 2^2 - 540; D(1) = 40 \ F^1 - 40 \\
D = D(0) + D(1) = 580 \\
c(0) = \text{round}(100 \times 0 + 100 \times 1 + 100 \times 2 + 40 \times 3) / 340 = 1 \\
c(1) = \text{round}(30 \times 4 + 20 \times 5 + 10 \times 6 + 0 \times 7) / 100 = 4 \\
\]

CSE 498gr - Lecture 11 - Winter 2004

Example

\[
\begin{array}{cccccccc}
\text{pixel value} & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\
\text{frequency} & 100 & 100 & 100 & 40 & 30 & 20 & 10 & 0 \\
\end{array}
\]

\[
c(0) = 1; c(1) = 5 \\
X'(0) = \{0.2\}, X'(1) = \{3.7\} \\
D'(0) = 200 \ F^1 + 200 \\
D'(1) = 40 \ F^1 + 40 \ 2^2 = 200 \\
D' = D'(0) + D'(1) = 400 \\
\]

\[
c = c' = X' = D' = X' \\
\]

CSE 498gr - Lecture 11 - Winter 2004
Scalar Quantization Notes

- Needed for analog to digital conversion.
- Useful for estimating a large set of values with a small set of values.
- With entropy coding yields good lossy compression.
- Lloyd algorithm works very well in practice, but can take many iterations.
  - For n codewords should use about 20n size representative training set.
  - Imagine 1024 codewords.