Lossy Image Compression Methods

- Scalar quantization (SQ).
- Vector quantization (VQ).
- DCT Compression
  - JPEG
- Wavelet Compression
  - SPIHT
  - GTW
  - EBCOT

JPEG Standard

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

Barbara

32:1 compression ratio
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 25-30 dB

Rate-Fidelity Curve

Properties:
- Increasing
- Slope decreasing
PSNR is not Everything

PSNR = 25.8 dB

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

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

Example

- 512 x 512 image = 216,144 pixels

<table>
<thead>
<tr>
<th>Index</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>9,144</td>
</tr>
<tr>
<td>1</td>
<td>18,000</td>
</tr>
<tr>
<td>2</td>
<td>10,000</td>
</tr>
<tr>
<td>3</td>
<td>10,000</td>
</tr>
<tr>
<td>4</td>
<td>10,000</td>
</tr>
<tr>
<td>5</td>
<td>10,000</td>
</tr>
<tr>
<td>6</td>
<td>25,000</td>
</tr>
<tr>
<td>7</td>
<td>9,144</td>
</tr>
</tbody>
</table>

Huffman Tree

- $A R^B = (100000 \times 1 + 90000 \times 2 + 43000 \times 4 + 38144 \times 5)/216144 \approx 2.997$
- Arithmetic coding should work better.

Improving Distortion

- Choose the codeword as a weighted average

$$c(j) = \text{round}(\sum_{x \in \text{index}(j)} x \cdot p_i)$$

$\text{Let } p_i \text{ be the probability that a pixel has value } x.$

$\text{Let } [L, R) \text{ be the input interval for index } j.$

$c[j] \text{ is the codeword indexed } j.$
Example

All pixels have the same index.

<table>
<thead>
<tr>
<th>Pixel Value</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>9</td>
</tr>
<tr>
<td>1</td>
<td>9</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
</tr>
<tr>
<td>3</td>
<td>11</td>
</tr>
<tr>
<td>4</td>
<td>12</td>
</tr>
<tr>
<td>5</td>
<td>13</td>
</tr>
<tr>
<td>6</td>
<td>14</td>
</tr>
<tr>
<td>7</td>
<td>15</td>
</tr>
</tbody>
</table>

New Codeword = round((8 * 9 + 10 + 100 + 11 + 10 + 13 + 12 + 14) / 15) = 10
Old Codeword = 11

New Distortion = 140 * 1^2 + 130 * 2^2 + 80 * 3^2 + 10 * 4^2 = 16000
Old Distortion = 130 * 1^2 + 120 * 2^2 + 110 * 3^2 = 16000

An Extreme Case

Frequency of pixel values

Only two codewords are ever used!!

Non-uniform Scalar Quantization

Frequency of pixel values

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
  \[
  D = \sum p_x (x - c(\text{index}(x)))^2
  \]
- Goal of the Lloyd algorithm is to find the codewords that minimize distortion.
- Lloyd finds a local minimum by an iteration process.

Example

Initially \( c(0) = 2 \) and \( c(1) = 5 \)

<table>
<thead>
<tr>
<th>Pixel Value</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>100</td>
</tr>
<tr>
<td>1</td>
<td>100</td>
</tr>
<tr>
<td>2</td>
<td>100</td>
</tr>
<tr>
<td>3</td>
<td>40</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>6</td>
<td>7</td>
</tr>
</tbody>
</table>

\( X(0) = [0.3], X(1) = [4.7] \)
\( D(0) = 140 * 1^2 + 100 * 2^2 = 540 \)
\( D(1) = 40 * 1^2 = 40 \)
\( D = D(0) + D(1) = 580 \)
\( c(0) = \text{round}(100 * 0.3 + 100 * 2 + 40 * 0.3) = 1 \)
\( c(1) = \text{round}(30 * 4 + 20 * 5 + 10 * 6 + 0.7 * 0.3) = 5 \)
### Example

<table>
<thead>
<tr>
<th>Pixel Value</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>100</td>
</tr>
<tr>
<td>1</td>
<td>100</td>
</tr>
<tr>
<td>2</td>
<td>100</td>
</tr>
<tr>
<td>3</td>
<td>100</td>
</tr>
<tr>
<td>4</td>
<td>60</td>
</tr>
<tr>
<td>5</td>
<td>30</td>
</tr>
<tr>
<td>6</td>
<td>20</td>
</tr>
<tr>
<td>7</td>
<td>10</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
</tr>
</tbody>
</table>

- \(c(0) = 1 \), \(c(1) = 5\)
- \(X(0) = [0,2], X(1) = [3,7]\)
- \(D(0) = 200, \Delta = 200\)
- \(D(1) = 40, \Delta = 40\)
- \(D = D(0) + D(1) = 400\)
- \(D - D(1) = 400 - 400/580 = 0.31\)
- \(c = c = X, D = D\)

```
c(0) = 1; c(1) = 5
X(0) = [0,2]; X(1) = [3,7]
D = 400
D(0) = round((100 * 0 + 100 * 1 + 100 * 2) / 300) = 1
D(1) = round((40 * 3 + 30 * 4 + 20 * 5 + 10 * 6 + 0.7) / 100) = 4
```

### Example

<table>
<thead>
<tr>
<th>Pixel Value</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>100</td>
</tr>
<tr>
<td>1</td>
<td>100</td>
</tr>
<tr>
<td>2</td>
<td>100</td>
</tr>
<tr>
<td>3</td>
<td>100</td>
</tr>
<tr>
<td>4</td>
<td>60</td>
</tr>
<tr>
<td>5</td>
<td>30</td>
</tr>
<tr>
<td>6</td>
<td>20</td>
</tr>
<tr>
<td>7</td>
<td>10</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
</tr>
</tbody>
</table>

- \(c(0) = 1 \), \(c(1) = 4\)
- \(X(0) = [0,2], X(1) = [3,7]\)
- \(D(0) = 200, \Delta = 200\)
- \(D(1) = 60, \Delta = 10 * 2^2 = 100\)
- \(D = D(0) + D(1) = 400\)
- \(D - D(1) = 400 - 300 / 400 = 0.25\)
- \(c = c = X, D = D\)

```
c(0) = 1; c(1) = 4
X(0) = [0,2]; X(1) = [3,7]
D = 400
D(0) = round((100 * 0 + 100 * 1 + 100 * 2) / 300) = 1
D(1) = round((40 * 3 + 30 * 4 + 20 * 5 + 10 * 6 + 0.7) / 100) = 4
```

### Scalar Quantization Notes

- Useful 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.