CSE 490 GZ
Introduction to Data Compression
Winter 2002

Lossy Image Compression
Scalar Quantization

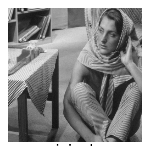
Lossy Image Compression Methods

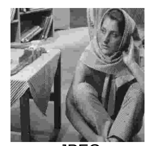
- Scalar quantization (SQ).
- Vector quantization (VQ).
- DCT Compression
  – JPEG
- Wavelet Compression
  – SPIHT
  – GTW
  – EBCOT

JPEG Standard

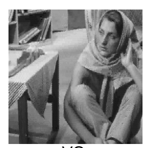
- JPEG - Joint Photographic Experts Group
  – Current image compression standard. Uses discrete cosine transform, scalar quantization, and Huffman coding.
- JPEG 2000 uses to wavelet compression.

Barbara


original


JPEG

32:1 compression ratio
.25 bits/pixel (8 bits)


VQ


Wavelet-SPIHT

JPEG

VQ

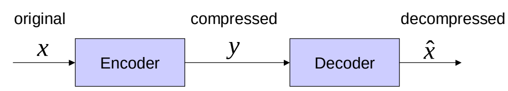
SPIHT

Original

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

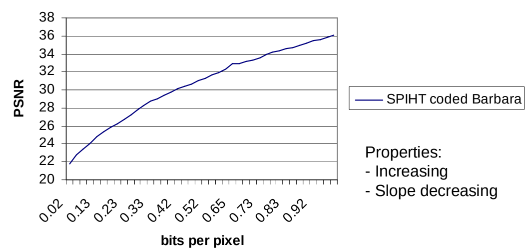
- Peak Signal to Noise Ratio (PSNR) is the standard way to measure fidelity.

$$PSNR = 10\log_{10}\left(\frac{m^2}{MSE}\right)$$

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

VQ

PSNR = 25.8 dB　　PSNR = 25.8 dB

PSNR Reflects Fidelity (1)

VQ

PSNR 25.8
.63 bpp
12.8 : 1

PSNR Reflects Fidelity (2)

VQ

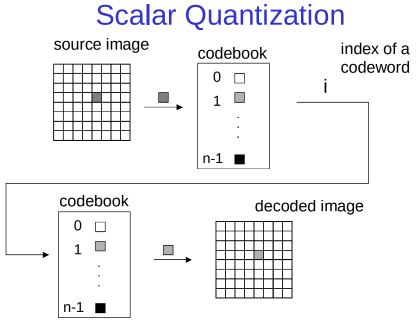
PSNR 24.2
.31 bpp
25.6 : 1

PSNR Reflects Fidelity (3)

VQ

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

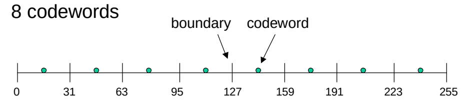
- Let the image be pixels $x_1, x_2, \ldots x_T$.
- Define index(x) to be the index transmitted on input x.
- Define c(j) to be the codeword indexed by j.

$$D = \sum_{i=1}^{T} (x_i - c(\text{index}(x_i)))^2 \qquad \text{(Distortion)}$$

$$MSE = \frac{D}{T}$$

Uniform Quantization Example

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



Codebook

Index	0	1	2	3	4	5	6	7
Codeword	16	47	79	111	143	175	207	239

Uniform Quantization Example

Encoder

input	0-31	32-63	64-95	96-127	128-159	160-191	192-223	224-255
code	000	001	010	011	100	101	110	111

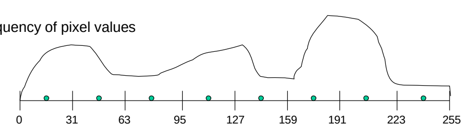
Decoder

code	000	001	010	011	100	101	110	111
output	16	47	79	111	143	175	207	239

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

Improving Bit Rate

Frequency of pixel values



q_j = the probability that a pixel is coded to index j
Potential average bit rate is entropy.

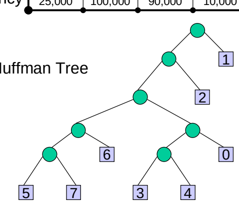
$$H = \sum_{j=0}^{7} q_j \log_2\left(\frac{1}{q_j}\right)$$

Example

- 512 x 512 image = 216,144 pixels

index	0	1	2	3	4	5	6	7
input	0-31	32-63	64-95	96-127	128-159	160-191	192-223	224-255
frequency	25,000	100,000	90,000	10,000	10,000	10,000	18,000	9,144

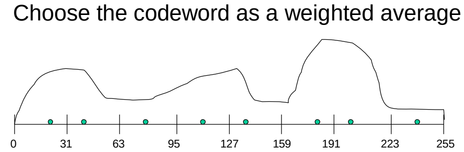
Huffman Tree



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

$$c(j) = \text{round}\left(\sum_{L_j \le x < R_j} x \cdot p_x\right)$$

Example

All pixels have the same index.

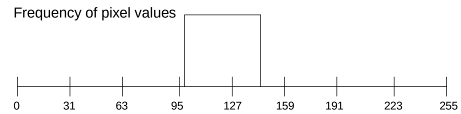
pixel value	8	9	10	11	12	13	14	15
frequency	100	100	100	40	30	20	10	0

$$\text{New Codeword} = \text{round}\left(\frac{8\cdot 100 + 9\cdot 100 + 10\cdot 100 + 11\cdot 40 + 12\cdot 30 + 13\cdot 20 + 14\cdot 10 + 15\cdot 0}{400}\right) = 10$$

$$\text{Old Codeword} = 11$$

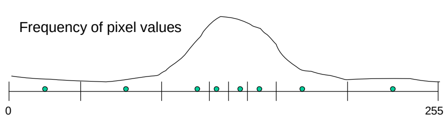
$$\text{New Distortion} = 140\cdot 1^2 + 130\cdot 2^2 + 20\cdot 3^2 + 10\cdot 4^2 = 10000$$
$$\text{Old Distortion} = 130\cdot 1^2 + 120\cdot 2^2 + 110\cdot 3^2 = 16000$$

An Extreme Case



Only two codewords are ever used!!

Non-uniform Scalar Quantization



- 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),...,c(n-1) and pixel x let index(x) be the index of the closest code word to x.
- Expected distortion is

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

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

Lloyd Algorithm

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

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

  Compute X'(j) = {x : x is a pixel value closest to c'(j)}
  Compute distortion D' for c'(0),c'(1),...,c'(n-1)
   if |(D – D')/D| < ε then quit
   else c := c'; X := X', D := D'
End{repeat}

Example

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

pixel value	0	1	2	3	4	5	6	7
frequency	100	100	100	40	30	20	10	0

$$X(0) = [0,3],\ X(1) = [4,7]$$
$$D(0) = 140\cdot 1^2 + 100\cdot 2^2 = 540;\ D(1) = 40\cdot 1^2 = 40$$
$$D = D(0) + D(1) = 580$$
$$c'(0) = \text{round}((100\cdot 0 + 100\cdot 1 + 100\cdot 2 + 40\cdot 3)/340) = 1$$
$$c'(1) = \text{round}((30\cdot 4 + 20\cdot 5 + 10\cdot 6 + 0\cdot 7)/60) = 5$$

Example

pixel value	0	1	2	3	4	5	6	7
frequency	100	100	100	40	30	20	10	0

$c'(0) = 1; c'(1) = 5$

$X'(0) = [0,2]; X'(1) = [3,7]$

$D'(0) = 200 \cdot 1^2 = 200$

$D'(1) = 40 \cdot 1^2 + 40 \cdot 2^2 = 200$

$D' = D'(0) + D'(1) = 400$

$|(D - D')/D| = (580 - 400)/580 = .31$

$c := c'; X := X'; D := D'$

Example

pixel value	0	1	2	3	4	5	6	7
frequency	100	100	100	40	30	20	10	0

$c(0) = 1; c(1) = 5$

$X(0) = [0,2]; X(1) = [3,7]$

$D = 400$

$c'(0) = \text{round}((100 \cdot 0 + 100 \cdot 1 + 100 \cdot 2)/300) = 1$

$c'(1) = \text{round}((40 \cdot 3 + 30 \cdot 4 + 20 \cdot 5 + 10 \cdot 6 + 0 \cdot 7)/100) = 4$

Example

pixel value	0	1	2	3	4	5	6	7
frequency	100	100	100	40	30	20	10	0

$c'(0) = 1; c'(1) = 4$

$X'(0) = [0,2]; X'(1) = [3,7]$

$D'(0) = 200 \cdot 1^2 = 200$

$D'(1) = 60 \cdot 1^2 + 10 \cdot 2^2 = 100$

$D' = D'(0) + D'(1) = 300$

$|(D - D')/D| = (400 - 300)/400 = .25$

$c := c'; X := X'; D := D'$

Example

pixel value	0	1	2	3	4	5	6	7
frequency	100	100	100	40	30	20	10	0

$c(0) = 1; c(1) = 4$

$X(0) = [0,2]; X(1) = [3,7]$

$D = 400$

$c'(0) = \text{round}((100 \cdot 0 + 100 \cdot 1 + 100 \cdot 2)/300) = 1$

$c'(1) = \text{round}((40 \cdot 3 + 30 \cdot 4 + 20 \cdot 5 + 10 \cdot 6 + 0 \cdot 7)/100) = 4$

Example

pixel value	0	1	2	3	4	5	6	7
frequency	100	100	100	40	30	20	10	0

$c'(0) = 1; c'(1) = 4$

$X'(0) = [0,2]; X'(1) = [3,7]$

$D'(0) = 200 \cdot 1^2 = 200$

$D'(1) = 60 \cdot 1^2 + 10 \cdot 2^2 = 100$

$D' = D'(0) + D'(1) = 300$

$|(D - D')/D| = (300 - 300)/300 = 0$

Exit with codeword $c(0) = 1$ and $c(1) = 4$.

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.