Image Compression

OUTLINE:

- Exploiting coding redundancy, interpixel redundancy, and psychovisual redundancy
- Lossy and lossless methods

Image Compression

Pictures take up a lot of storage space (either disk or memory).

A 1000x1000 picture with 24 bits per pixel takes up 3 megabytes.

The Encyclopaedia Brittanica scannned at 300 pixels per inch and 1 bit per pixel requires 25,000 pages $\times$ 1,000,000 bytes per page = 25 gigabytes.

Video is even bulkier: 90 minute movie at 640$\times$480 resolution spatially, 24 bit per pixel, 24 frames per second, requires 90$\times$60$\times$24$\times$640$\times$480$\times$3=120 gigabytes.

Applications: HDTV, film, remote sensing and satellite image transmission, network communication, image storage, medical image processing, fax.

How can we save space?

Three approaches:

Reduce Coding Redundancy - some pixel values more common than others.

Reduce Interpixel Redundancy - neighboring pixels have similar values.

Reduce Psychovisual Redundancy - some color differences are imperceptible.

Trade Off Quality Against Compression

Lossless compression (information preserving)

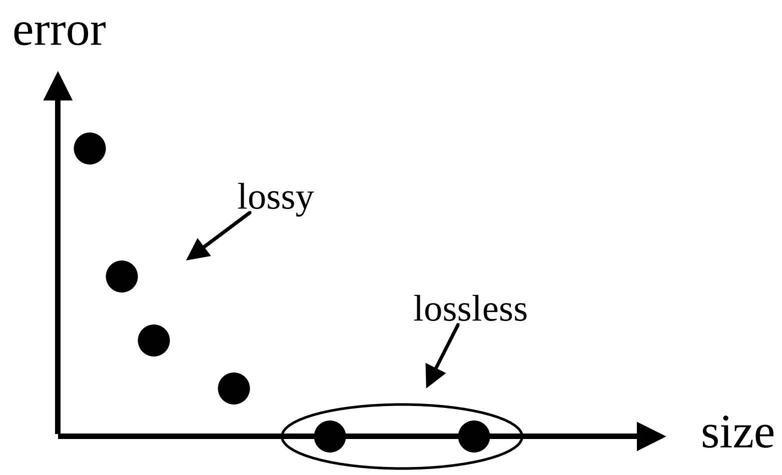
- original can be recovered exactly.

Higher quality, bigger.

Lossy compression

- only an approximation of the original can be recovered.

Lower quality, smaller.



Exploiting Coding Redundancy

These methods, from information theory, are not limited to images, but apply to any digital information. So we speak of “symbols” instead of “pixel values” and “sources” instead of “images”.

The idea: instead of natural binary code, where each symbol is encoded with a fixed-length code word, exploit nonuniform probabilities of symbols (nonuniform histogram) and use a variable-length code.

Entropy $H = -\sum \text{Prob}[symbol_i] \log_2 (\text{Prob}[symbol_i])$ is a measure of the information content of a source. If source is an independent random variable then you can’t compress to fewer than H bits per symbol.

Assign the more frequent symbols short bit strings and the less frequent symbols longer bit strings. Best compression when redundancy is high (entropy is low, histogram is highly skewed).

Two common methods: Huffman coding and LZW coding.

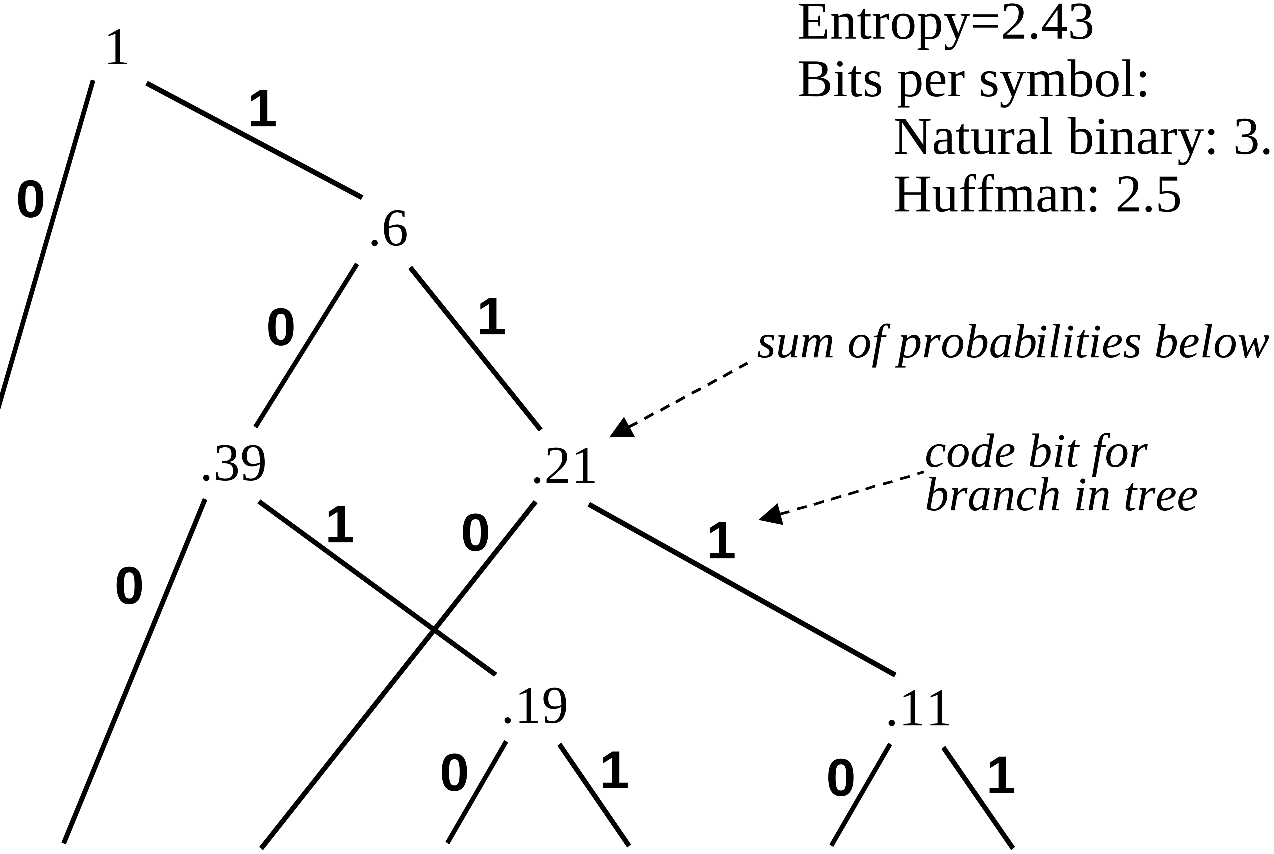
Huffman Coding

- Codebook is precomputed and static.
- Compute probabilities of each symbol by histogramming source.
- Process probabilities to precompute codebook: $code_i$.
- Encode source symbol-by-symbol: $symbol_i \rightarrow code_i$.

The need to preprocess the source before encoding begins is a disadvantage of Huffman coding.

Huffman Coding Example

Entropy=2.43
Bits per symbol:
Natural binary: 3.0
Huffman: 2.5



Prob$_i$	.4	.2	.1	.1	.09	.06	.05	0
symbol$_i$	010	100	111	000	101	001	011	110
code$_i$	**0**	**100**	**110**	**1010**	**1011**	**1110**	**1111**	-

Lempel-Ziv-Welch (LZW) coding

Codebook is computed on the fly (dynamic), an advantage over Huffman.

If source uses 2^k symbols, create table of length 2^j mapping strings to codes, $j >> k$.

(for 8-bit pixels, k=8, use j =12, perhaps)

Initialize table with the 2^k possible symbols.

$S \leftarrow$ “” (null string)

while there are more symbols in input

$P \leftarrow$ current symbol

if string SP is in table, $S \leftarrow SP$

else

output j-bit code for S

add SP to table if there is room

$S \leftarrow P$

output j-bit code for S

Exploiting Interpixel Redundancy, 1

Neighboring pixel values are similar and correlated, both in space and in time, i.e. pixels are not independent random variables.

Note: on certain images, the following methods cause expansion, not compression. Hybrid methods switch between basic methods depending on which gives best compression for a given scan line or image region.

Three spatial methods for low-noise images:

Run-Length Coding (RLC, RLE).

Output value1, run-length1, value2, run-length2, ...

Good compression if most horizontal neighbor pixels are the same.

Poor compression if picture is noisy. Can be done in 1-D or 2-D (the CCITT† FAX standard uses both).

Quadtrees: recursively subdivide until cells are constant color.

Good if picture has large regions of constant color with horz. & vert. boundaries.

Region Encoding: encode boundary curves of constant-color regions.

Good if picture has large regions of constant color.

† International Telegraph and Telephone Consultative Committee

Exploiting Interpixel Redundancy, 2

Three spatial methods that tolerate noise better:

- **Predictive Coding**: predict next pixel based on previous, output difference between actual and predicted.
- **Fractal Image Compression**: decribe image in terms of recursive affine transformations.
- **Transform Coding**: transform to a frequency or other domain where correlation is lower.
  - Examples: Discrete Cosine Transform (DCT), DFT, Karhunen-Loeve Transform.
  - Used in JPEG.

Temporal method:

- **Motion Compensation:** exploit interframe coherence.
  - Exploit similarity between corresponding pixels of successive frames of motion sequence. Used in MPEG.

JPEG & Discrete Cosine Transform

The discrete cosine transform (DCT) is frequently used for lossy compression. Like a DFT, but all real (uses cosines instead of complex exponentials).

$$F(u,v) = c(u)c(v)\sum_{x=0}^{N-1}\sum_{y=0}^{N-1} f(x,y)\cos\frac{(2x+1)u\pi}{2N}\cos\frac{(2y+1)v\pi}{2N}$$

where $c(u) = 1/\sqrt{N}$ if $u = 0$, $\sqrt{2/N}$ otherwise

JPEG (Joint Photographic Experts Group) still picture compression standard uses the following steps:

– subdivide image into $N \times N$ subimages (blocks), N=8 is common

– DCT each block

– quantize, zigzag order, and run-length code the coefficients

– use variable length coding (e.g. Huffman coding)

JPEG can compress many computer-generated pictures to 1-2 bits per pixel, and natural images to about 4 bits per pixel, with little visible error.

Goals of MPEG Digital Video Standard

MPEG (Motion Picture Experts Group) video compression standard had the following goals:

- VHS-quality video for 1.5 Mbits/sec
  - (a naive encoding might use $640 \times 480 \times 24 \times 30 = 221$ Mbits!)
- random access
- fast forward/reverse, play backward
- audio/video synchronization
- robust to errors
- editable
- flexible format (size & frame rate)
- real time encoder implementable in 1990 hardware

MPEG Digital Video Standard

MPEG compression steps build on JPEG:

- motion compensation using 16x16 pixel blocks
- discrete cosine transform (DCT) of 8x8 blocks
- quantization
- run-length coding of zig-zag scan of DCT coefficients
- Huffman coding

To exploit temporal coherence, three frame types:

I = Intrapicture

self-contained (a JPEG, basically) - *biggest, easiest to decode*

P = Predicted picture

described relative to previous I or P frame using motion vectors on image blocks, and JPEG-like coding of differences

B = bidirectional interpolation picture

interpolated from prevous and following I or P frames, similarly coded - *smallest, hardest to decode*

Typical frame sequence: **IBBBPBBB** (repeat)

Exploiting Psychovisual Redundancy

Exploit variable sensitivity of humans to colors:

We're more sensitive to differences between dark intensities than bright ones.

Encode log(intensity) instead of intensity.

We're more sensitive to high spatial frequencies of green than red or blue.

Sample green at highest spatial frequency, blue at lowest.

We're more sensitive to differences of intensity in green than red or blue.

Use variable quantization: devote most bits to green, fewest to blue.

NTSC Video

NTSC video exploits properties of human visual system to represent color pictures:

$$\begin{bmatrix} Y \\ I \\ Q \end{bmatrix} = \begin{bmatrix} .30 & .59 & .11 \\ .60 & -.28 & -.32 \\ .21 & -.52 & .31 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

Y bandlimited to 4.2 MHz

I to 1.6 MHz

Q to .6 MHz

NTSC is interlaced.

Further Reading

Gonzalez & Woods, *Digital Image Processing*
chapter on Image Compression

Netravali & Haskell, *Digital Pictures*

April 1991 issue of *Communications of the ACM*

Articles on recent work in wavelet image/video compression