Image Compression & Image Formats

Lecture 11

February 18, 2002

Formats, Formats

As you may already have discovered, there are dozens of common image formats. Why?

- Optimized for different image types
  - 2D/3D, bw/color, real/integer, sequences, range…
- History & corporate policy
  - GIF was proprietary, MPG is owned by a consortium

Dull

Interesting

- Compression Schemes
  - LUT, RLE, LZW, DCT

n

Does it matter?

- Limited Transmission Bandwidth
  - videoconferencing
  - the Internet
- Limited Storage Capabilities
  - Imagine trying to store the images off your cable system:
    - ~100 stations
    - 30 frames/sec
    - 3 colors
    - ~600x400 pixels per frame
    - Total: over 2 GPixels per second!
  - DVDs
- Limited retrieval from storage
  - FBI mug shot database (too many faces to compare!)

Simple, lossless compression...

Color Maps (LUTs)

Lets assume that the source image is made up of 3 8-bit (RGB) values per pixel. Then..

- If the image is smaller than 4,000 x 4,000, not every color combination is used
  - Even if it is larger, some combinations are probably missed…
- This can be used to compress the data:
  - store a table of the used colors (any order)
  - for every pixel, store the table index of the corresponding color

Q: When does this compress data?

More simple, lossless compression...

Run Length Encoding (RLE)

Another simple trick is to note that pixels often have the same value as the pixel next to them.

- RLE: Values are written as pairs:
  - the pixel value
  - the number of consecutive pixels with this value
    - in scan-line order
- Generally compresses
  - Works very well for flat graphics (e.g. icons)
  - Can expand rather than compress a highly textured image
  - Works well in combination with color maps

GIF

- GIF is a format developed back when the only images that PCs could manipulate were simple graphics.
  - Employs a color table
    - Users can set it to 8, 16, or 24 bytes
  - Employs run length encoding
  - Other, irrelevant oddities (like the coordinate system)
- GIF compresses “flat” graphics very well
- Compresses most images very poorly
- New versions have a lot of settings that applications programmers can control...

Lempel-Ziv-Welch (LZW)

LZW is a string coding algorithm. Lets assume 8-bit source data (black and white, this time)

- LZW uses code words that are longer than the pixel size
  - for 8-bit data, 10-bit codes are common
- A table is initialized with all possible pixel values
  - e.g. a 10-bit table, with values 0-255 pre-assigned.

LZW (II)

- Now, the trick is to encode sequences:
  - if the current pixel and the next pixel are not in the table as a pair, add them to the table and store the index.
  - If they are in the table, consider them as one pixel and see if that plus the next neighbor is in the table as a pair; if not, add & store index
  - When table is full, record index of longest entry starting at the current pixel

LZW (Example)

Original Image

11 34 11 34 35 34 35 38 35 36 37 38 39 34 35 40...

…	…
254	254
255	255
256	11, 34
257	256, 35
258	34, 35
259	38, 35
260	36, 37
261	38, 39
262	258, 40

Compressed Image

256 257 258 259 260 261 262...

TIFF

- LZW is the backbone of the TIFF file format
- It is also an option for GIF
- This may or may not be legal, since Unisys claims to have a patent on the LZW algorithm.
- TIFF can also be blocked to aid locality of reference
- TIFF is the best lossless format for general images
- An API library for the tiff format is freely available

Huffman Coding

Huffman coding is an optimal, lossless compression scheme

- Every pixel is mapped onto a variable-length bit string according to a probability table, as follows:
  - For an example, I will show an example of Huffman coding digits (0-9), but for an 8-bit image you would Huffman code 0-255

Digit	0	1	2	3	4	5	6	7	8	9
Prob.	.1	.3	.1	.03	.1	.05	.1	.06	.05	.1

Digit	0	1	2	4	6	7	8	9	3/5
Prob.	.1	.3	.1	.1	.1	.06	.05	.1	.08

Huffman (II)

Digit	0	1	2	4	6	9	3/5	7/8
Prob.	.1	.3	.1	.1	.1	.1	.08	.11

Digit	0	1	2	4	6	3/5/9	7/8
Prob.	.1	.3	.1	.1	.1	.18	.11

Digit	0	1	2	4/6	3/5/9	7/8
Prob.	.1	.3	.1	.2	.18	.11

Digit	1	0/2	4/6	3/5/9	7/8
Prob.	.3	.2	.2	.18	.11

Digit	1	0/2	4/6	3/5/7/8/9
Prob.	.3	.2	.2	.29

Huffman (III)

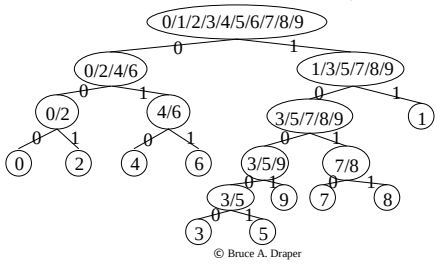
Digit	1	0/2/4/6 3/5/7/8/9
Prob.	.3	.4 .29

Digit	0/2/4/6 1/3/5/7/8/9
Prob.	.4 .59

Digit	0/1/2/3/4/5/6/7/8/9
Prob.	.99

Huffman (IV)

Now arrange in a tree; the code for every symbol is its path
(Note that these are variable length)



Huffman (V)

- Once the table is built, each pixel is converted to a bit string, and all the bit strings are concatenated.

Digit	0	1	2	3	4	5	6	7	8	9
Code	000	11	001	10000	010	10001	011	1010	1011	1001

Discrete Cosine Transform (DCT)

Intuitive Version Only

- The Discrete Cosine Transform is similar to the Fourier transform, in that is represents a signal as a sum of cosine waves.
- It differs in that it mirrors the signal (image) in both directions, fixing the phase of the cosines
- The DCT therefore has no imaginary component
- The size of the DCT description of an image is the same as the size of the original image

A lossy compression standard

JPEG

- JPEG first divides the image into 8x8 subimages
- For each subimage, it:
  - Computes the DCT (limited to 11 bits, assuming 8 bit source)
  - DC component is difference coded to previous 8x8 square
  - For every compression "quality level" (1-100), it assigns a number of bits per frequency
  - Quantizes the frequency information
  - Unfolds 2D array in S pattern to form 1D vector
  - Computes a Huffman code for the discretized frequencies.
- This is the best widely-available "lossy" standard

How about video?

MPEG

- MPEG is a compression for motion video
  - Actually, it's a series of them: MPG1, MPG2,…
- It works by matching an image to its successor:
  - compute the transformation that best maps the image onto its successor
  - encode the successor as the difference between the transformed source and the true successor
  - since the result is mostly zeroes, it compresses well
    - via Huffman, LZW, or RLE
- But to understand the details, we have to know how to match images….