CS559 – Image Representation and Compression

December 2007

Notes for reference, not for projection

Image File Formats

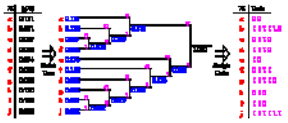
- Need to store all of the samples
- At whatever the necessary bits per pixel
- Lots of data
- Uncompressed = big
- Compress to take less space
  - Lossless (get same thing out)
  - Lossy (lose some information)

Lossless Coding 1

- Run-Length Encoding (RLE)
- Send pairs of values/run lengths
  - Only a win if (on average) your runs are long
- Look ahead:
  - Small change can mean big difference in coding
  - What if the changes were small enough that no one notices?

Lossless Coding 2

- Intelligent coding – give short codes to more common strings
  - Example: letters – rather than each getting 8 bits, let E=10, A=00, T=001, …
  - If you know the frequency distribution, you can distribute things optimally – Huffman encoding
  - Optimal Distribution may be uniform!
  - Entropy: the amount of distribution in the data
- Some things can't be made smaller by lossless encoding



Entropy Coding

- Fixed / Variable sized strings for codes
- Standard Codebook vs. per-corpus (file/image)
- Many algorithms for doing this
  - Huffman coding is just one classic one
- Lempel-Ziv (or Ziv-Lempel)
  - Variable length strings
  - Fixed code sizes (all the same)

Lossless Image Compression

- Use entropy coding (like LZ) on the actual pixels
- File formats
  - GIF – patented, only for small color paletes
  - PNG
- Uncompressed (or optionally compressed)
  - TGA (targa)
  - TIFF
  - BMP

Lossy Image Compression

- What if we limit our codebook?
  - Some data cannot be represented exactly
- Vector Quantization
  - Fixed length strings (and fixed codebook size)
  - Pick a set of codes that are as good as possible
  - Encode data by picking closest codes
  - Other than picking codes, encoding/decoding is really easy!

Lossy Coding 2

- Suppose we can only send a fraction of the image
  - Which part?
- Send half an image:
  - Send the top half (not too good)
  - Halve the image in size (send the low frequency half)
- Idea: re-order (transform) the image so the important stuff is first

Lossy Coding 2

- Suppose we can only send a fraction of the image
  - Which part?
- Send half an image:
  - Send the top half (not too good)
  - Halve the image in size (send the low frequency half)
- Idea: re-order (transform) the image so the important stuff is first

Perceptual Image Coding

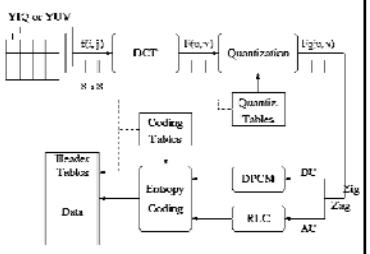
- Idea: lose stuff in images that is least important perceptually
  - Stuff least likely to notice
  - Stuff most likely to convey image
- Who knows about this stuff: The experts!
  - Joint Picture Experts Group
  - Idea of perceptual image coding

JPEG

- Key Ideas
  - Frequency Domain (small details are less important)
  - Block Transforms (works on 8x8 blocks)
    - Discrete Cosine Transform (DCT)
  - Control Quantization of frequency components
    - More quality = use more bits
    - Generally, use less bits for HF

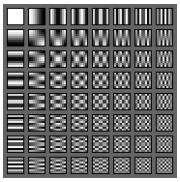
JPEG

- Multi-stage process intended to get very high compression with controllable quality degradation
- Start with YIQ color
  - Why? Recall, it's the color standard for TV



Discrete Cosine Transform

- A transformation to convert from the *spatial* to *frequency* domain – done on 8x8 blocks
- Why? Humans have varying sensitivity to different frequencies, so it is safe to throw some of them away
- Basis functions:



Quantization

- Reduce the number of bits used to store each coefficient by dividing by a given value
  - If you have an 8 bit number (0-255) and divide it by 8, you get a number between 0-31 (5 bits = 8 bits – 3 bits)
  - Different coefficients are divided by different amounts
  - Perceptual issues come in here
- Achieves the greatest compression, but also quality loss
- "Quality" knob controls how much quantization is done

Entropy Coding

- Standard lossless compression on quantized coefficients
  - Delta encode the DC components
  - Run length encode the AC components
    - Lots of zeros, so store number of zeros then next value
  - Huffman code the encodings

Lossless JPEG With Prediction

- Predict what the value of the pixel will be based on neighbors
- Record error from prediction
  - Mostly error will be near zero
- Huffman encode the error stream
- Variation works really well for fax messages

Video Compression

- Much bigger problem (many images per second)
- Could code each image seperately
  - Motion JPEG
  - DV (need to make each image a fixed size for tape)
- Need to take advantage that different images are similar
  - Encode the Changes ?

MPEG

- Motion Picture Experts Group
  - Standards organization
- MPEG-1 simple format for videos (fixed size)
- MPEG-2 general, scalable format for video
- MPEG-4 computer format (complicated, flexible)
- MPEG-7 future format
- What about MPEG-3? – it doesn't exist (?)
  - MPEG-1 Layer 3 = audio format

MPEG Concepts

- Keyframe
  - Need something to start from
  - “Reset” when differences get too far
- Difference encoding
  - Differences are smaller/easier to encode than images
- Motion
  - Some differences are groups of pixels moving around
  - Block motion
  - Object motion (models)

MPEG