DATA COMPRESSION – a quick introduction

Terminology

- **Un-encoded data** - raw data
- **Encoded data** - compressed data
- **Compression ratio** - raw data / compressed

Classification

Physical vs logical

- The terms squeezing, crunching, imploding are not very accurate because we do not store the same info in smaller spaces.
- However how we transform the data can be thought of as physical and logical, most good methods are physical.

Example of logical compression: Abbreviations

- United States of America becomes USA
- Only physical is used with images

Symmetrical vs Asymmetrical

- **Symmetrical**: same time to compress as to decompress

Adaptive

- **Adaptive** - Adaptive is most general
- **Semi-adaptive** - Semi makes two passes
- **Non-adaptive** - Non-adaptive only works with a given string, ex. Huffman codes

Lossy vs Lossless

Lossless methods

- Run Length Encoding
- Fax machines
- variations which direction to run along X axis, along Y axis, tiles(4x4 pixels?),
    - zig zag (diagonal changing directions)
- bit level, byte level, pixel level

Lempel Ziv

- Abraham Lempel, Jakob Ziv 1977
- used in compress, pkzip, GIF, TIFF
- modified by Terry Welch 1984 LWZ compression

- Uses fixed-length codewords to represent variable-length strings of symbols/characters that commonly occur together.

- The encoder and decoder both build the same dictionary dynamically while scanning the data.

- Builds its own dictionary of compressed values At each step the longest prefix ( ) of the remaining source string that matches an existing code in the table is pulled of with the next character (c) creating table entry (, c). The table represents the string.

Example:

aa_bbb_cccc_ddddd_eeeeee_fffffffggggggggg *where _ is blank*

(0,a) (1,_) (0,b) (3,b) (0,_) (0,c) (6,c) (6,_) (0,d) (9,d) (10,_) (0,e) (12,e) (13,e) (5,f) (0,f) (16,f) (17,f) (0,g) (19,g) (20,g) (20)

For text, play a trick, assume first 256 in table are already defined but don't store

In practice, the code length l is kept in the range of $[l_0, l_{max}]$ The dictionary initially has a size of 2 to the l_0 power. When it is filled, the code length will be increased by 1 until $l = l_{max}$

<u>Differential Encoding</u>

Particularly good with images

Given an original image I(x,y) use a simple difference operator to create

$$d(x,y) = I(x,y) - I(x-1,y)$$

or use the discrete version of the 2-D Laplacian operator to define a difference image d(x,y)

$$d(x,y) = 4I(x-y) - I(x,y-1) - I(x,y+1) - I(x+1,y) - I(x-1,y)$$

spatial redundancy

Lossy Compression Techniques

- Lossless compression algorithms often do not deliver compression ratios that are high enough. Most multimedia compression algorithms that are regularly used are lossy.
- The compressed data is not the same as the original data, but a close approximation of it.
- Usually yields a much higher compression ratio than that of lossless compression

Quantization – the heart of almost all lossy algorithms

- Reduce the number of distinct output values to a much smaller set.
- This is the main source of the "loss"

Uniform Scalar Quantization

- Partitions the domain of input values into equally spaced intervals, except possibly at the two outer intervals.
- Basically rounding with variations

Nonuniform

- If the data is not uniformly distributed, then the uniform quantizer does not work as well

Discrete Cosine Transformation (DCT)

- Basic idea is very similar to Fast Fourier Transform
- We can take any wave form and represent it as a sum of sine or cosine wave forms of different frequencies. To be exact, we usually need an infinite number of frequencies. To get close, we need very few - so drop the extra ones.
- How does this apply to sound and images?