Outline: Image Compression

		OUTPUT	
		Image	Description
INPUT	Image	Image Processing	Pattern Recognition
	Description	Computer Graphics	Data Processing

- Why compression?
- Lossless compression.
- Lossy compression.

Why compression?

- A 1024 $\times$ 768 pixel image in true colour (24 bit), would take about 2.4 Mb to store. Communicating this is where most of the problems arise.
- To the human eye there is a lot of redundant information in most images.
- We can consider two types of compression:
  - lossless;
  - lossy.
- Which compression algorithm to use depends on the image.

Lossless compression

- Rather like that used in `zip` and `gzip`.
- The uncompressed data is identical to the original data.
- When considering bitmap images many compression methods use scan line coherence.
- For colour images it is generally better to compress in HSV space rather than RGB space.
- Amount of compression achievable is related to image complexity, which can be measured in terms of entropy.

Run length encoding

Original Image

63	63	63	63	64	64	64	78	89	89	89	89

Compressed Image

63,4,64,3,78,1,89,4

- Code the number of pixels taking the same value along a given scan line.
- Works particularly well on binary images since only length of run needs to encoded.
- Works by utilising scan line coherence.

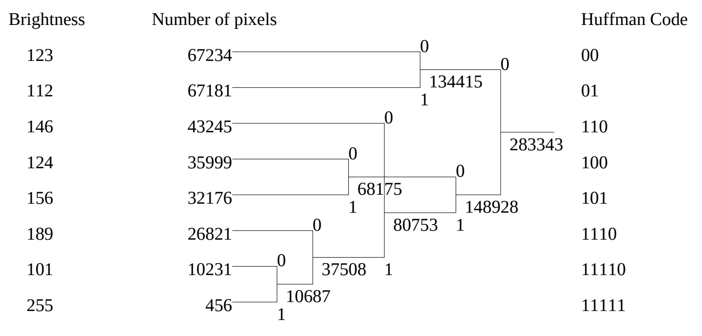
Run length encoding

- Bit-plane run length encoding is used on non-binary images by considering each bit of the, say 8 bit, image one at a time.
- Compression rates of 1.5:1 (gray-scale / colour images), 4:1 (binary images) and 2:1 (bit-plane compression on gray-scale / colour images)
- May cause a data explosion: the final file may be larger than the original one.

Huffman coding

- Huffman coding works on the image brightness histogram.
- Finds the most commonly occuring brightness patterns and uses the shortest codes to represent these.
- Compression rates of 1.5 – 2:1.
- Huffman coding may also be used after run length coding to give further compression.

An Example of Huffman coding



Predictive coding

Original Image

63	63	63	63	64	64	64	78	89	89	89	89

Compressed Image

63,0,0,0,1,0,0,14,11,0,0,0

- Stores the difference between successive pixels' brightness in fewer bits.
- Relies on the image having smooth changes in brightness: at sharp changes in the image we need overload patterns.
- Gives up to 2:1 image compression rates – can be improved by iterative application.

Block coding

- Block coding looks for frequently occuring block patterns in the image.
- Exploits object coherence and repetition in images – not just scan line coherence.
- More computationally expensive and need to send the codebook with the compressed image.
- Lemple-Ziv-Welch coding uses a variant of the block method:
  - Fix number of blocks, find the largest, most commonly occuring blocks (overload can be used).
  - Huffman code.

Block coding

- `.gif` file format uses a version of Lemple-Ziv-Welch coding.
- Typical compression rates of 2 – 3:1.
- For very low entropy images, such as those generated by simple computer graphics, much better compression rates of 10:1 can be achieved.
- Problem with lossless compression – image size still about 0.8 Mb at best.

Lossy image compression

- Accept a small loss of information from the image to achieve a better compression rate.
- Truncation coding is the simplest lossy compression method.
- Remove least significant bits (typically 2:1) or pixels (typically 4:1).
- Reconstruct pixel reduced image by interpolation.
- Reconstruct bit reduced image by adding dither noise to avoid posterisation.
- A dumb compression method.

Lossy predictive coding

Original Image

63	63	63	63	64	64	64	78	89	89	89	89

Compressed Image

63,0,0,0,1,0,0,8,8,8,1,0

Reconstucted Image

63	63	63	63	64	64	64	72	80	88	89	89

- Like predictive coding but no code overload.

Lossy predictive coding

- Delta modulation uses only one bit to indicate brightness difference $-1 = +1$, $0 =$ -1.
- Clearly not going to well reproduce most images .... but ....
- Compression rates of 3:1 to 8:1.
- Rarely used in practice.

Lossy block coding

- Again like its lossless counterpart.
- Uses codebook to reconstruct approximate image - thus can have a smaller codebook.
- The codebook is chosen to minimise the squared reconstruction error (or some other measure e.g. brightness difference).
- Mathematically involved.
- Fractal compression uses basic codebook blocks, then translations, rotations and scalings of these base blocks.
- Gives compression rates of 10:1 and better.

Transform coding

- Like block coding, but using fixed basis functions which thus are not transmitted.
- Most common is the discrete cosine transform (discrete version of the Fourier transform).
- The underlying assumption is that noise is in the high frequency components which are thrown away.
- The wavelet transform can be tuned to the images to be compressed.
- Gives typical compression rates of 10:1 and better with little loss of detail.

Transform coding

- `.jpg` file format uses:
  - the discrete cosine transform
  - followed by lossless predictive coding of the retained components
  - and then Huffman coding of the predictive coded image.
- Works on images with larger entropy.

Which to choose?

- Main choice is between lossy and lossless compression.
- Always a tradeoff between:
  - compression rates,
  - accuracy of reproduction,
  - processing requirements.
- Can go very statistical using the concept of entropy.

Summary

- Having finished this lecture you should:
  - understand the difference between lossy and lossless compression;
  - be able to implement some simple lossy and lossless compression algorithms;
  - discuss the entropy of different images and its effect on compression;
  - contrast the merits of different algorithms.
- Understanding the different compression methods should help you to decide which is most appropriate when.