IMAGE PROCESSING (RRY025)

LECTURE 13

IMAGE COMPRESSION - I

Need For Compression

- 2D data sets are much larger than 1D. TV and movie data sets are effectively 3D (2-space, 1-time).

- Need Compression for both transmission and storage.

- Applications

  - JPEG on internet

  - Picture telephones/video conferencing

  - Large image data bases (in US FBI stores 20 million fingerprints)

Types of Compression

- Applications to storage and transmission essentially the same. Although in the latter case may have to worry about transmission channel errors.

- *Lossless* compression is information preserving. - so is perfectly reversible. But often achievable compression is only modest (5:1 for grayscale images). Useful for permanent storage.

- *Lossy* compression does not preserve information. Can not be exactly reversed. Can get high compression, i.e. 20:1. Most useful for 'throw-away' applications such as browsing a web page.

Sources of Compression

- Compression works by removing *redundancy* from data.

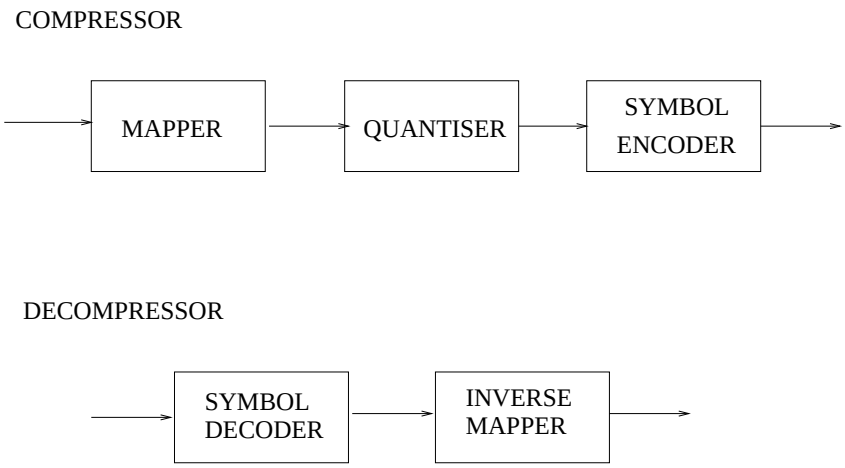
  – *Coding Redundancy.* Lossless. Give common values short codes and uncommon values long codes.

  – *Interpixel Redundancy.* Lossless/Lossy. Adjacent pixel values are highly correlated, so they don't send independent information.

  – *Psychovisual Redundancy.* Lossy. Not all (high spatial frequency) information is perceived by eye/brain so don't transmit.

- A highly redundant image (uniform gray) is highly compressible. Characterised by one number. Random pixel-to-pixel uncorrelated noise with uniform distribution is incompressible.

General Compressor/Decompressor



Mapper. Converts to a new (more compressible) image representation. i.e. take difference between pixels, convert to run lengths, or do a cosine transform.

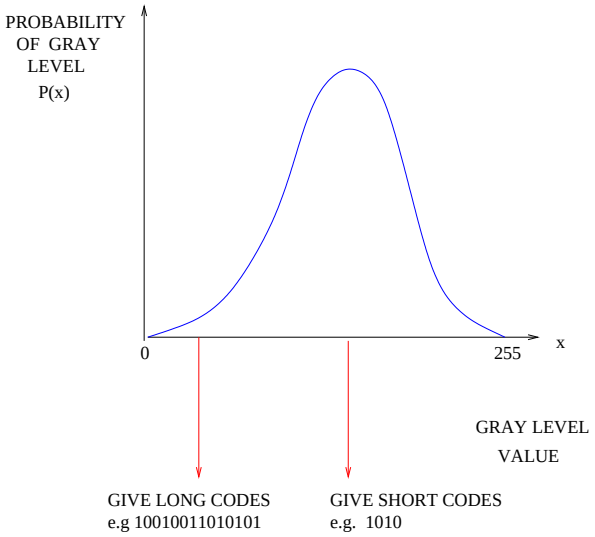
Quantiser. Really a 're-quantiser'. Take pixel values represented by 8 bits, lose accuracy and represent with 6 bits, therefore compress by factor $8/6 = 1.33$. *always lossy*.

Symbol Encoder. Encode common symbols with short codes and uncommon ones with long codes *lossless*

The decompresser consists of an inverse symbol encoder and inverse mapper. There is no inverse quantiser since that step is lossy and irreversible.

Symbol Coder

- Encode separately each of the pixel values. Give common pixel values short codes, rare values long codes. Coder and decoder use same 'codebook'.



- If the codeword for gray level value x has $N(x)$ bits then the mean number of bits per pixel of compressed data

$$N_{mean} = \sum_{o}^{255} P(x)N(x)$$

For the right codebook we can get $N_{mean} < 8$ and hence compression.

Coding Theorem

- 'Shannons Noiseless Coding Theorem' - it is possible to encode *without distortion* a sequence of single symbols of entropy H_1 per symbol with a code using $H_1 + \eta$ bits/symbol where η is arbitrarily small and where H_1 is;

$$H_1 = -\sum_{x=0}^{L-1} P(x) Log_2 P(x)$$

where L is the number of gray levels and $P(x)$ the probability of each gray level.

- The theorem however does not tell us how to construct the codebook. A simple way to approximate to it is via Huffman coding, where input pixel values of fixed length (say 8bits/pixel) are mapped to a variable length codewords.

  How do we know when the code for one pixel ends? Sending special combinations of bits to indicate pixel boundaries reduces compression. Instead make codewords *uniquely decodable*.

Huffman Coding

Construct according to the following rules.

- Arrange the symbol probabilities, $P(x)$ in decreasing order and consider them as leaf nodes on a tree.

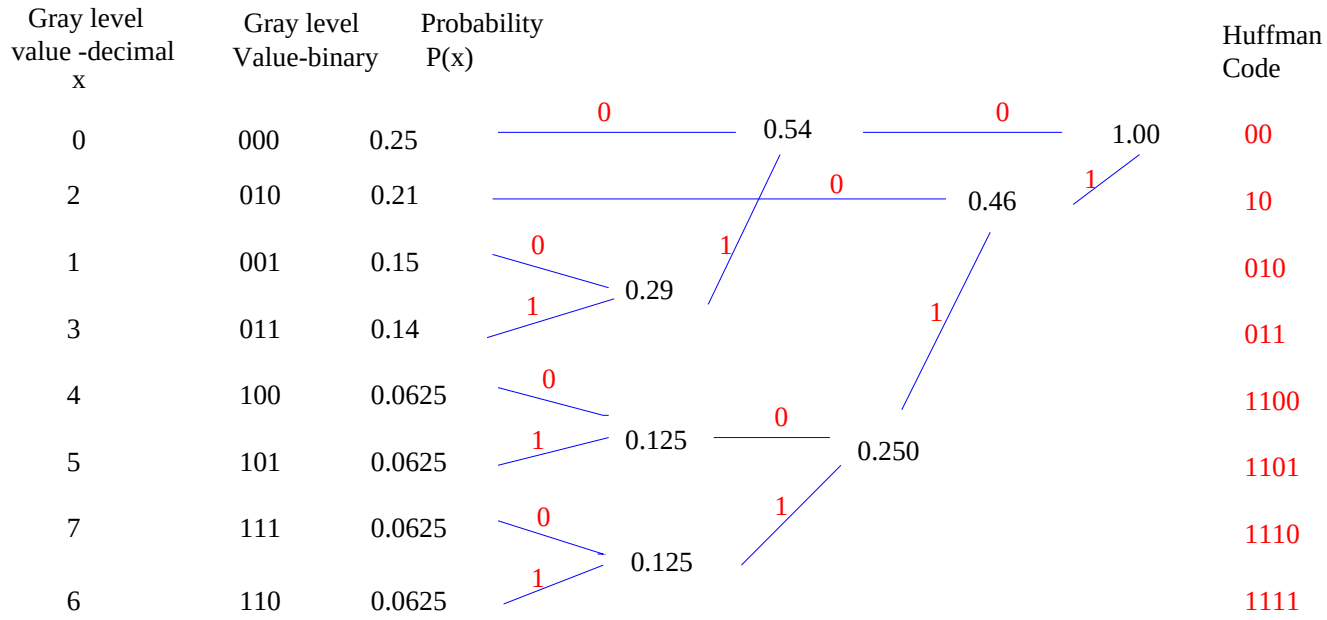
- While there is more than one node:

  - Merge the two notes with smallest probability to form a new node whose probability is the sum of the merged nodes.

  - Arbitrarily assign 1 or 0 to each pair of branches into a node (often assign 1 to the branch to the larger probability, but it doesn't matter)

- create the code for each symbol by reading sequentially from the root node to the leaf node.

- An example Huffman code for an image with 8 gray levels.



- This is uniquely decodeable sequence because the first n bits of any length m Huffman code (where n $<$ m) never equals another Huffman code. Can transmit coded data as a bit stream with no gaps.So the transmitted sequence 110000101101101111011 can be uniquely decoded as

  1100/00/10/1101/10/1111/011/

  or

  4 0 2 5 2 6 3

- For the above example we have

x	Input	Prob P(x)	Huffman Code	Code Length N(x)
0	000	0.25	00	2
2	010	0.21	10	2
1	001	0.15	010	3
3	011	0.14	011	3
4	100	0.0625	1100	4
5	101	0.0625	1101	4
7	111	0.0625	1110	4
6	110	0.0635	1111	4

- After encoding the mean length per pixel is

$$N_{mean} = \sum_{x=0}^{7} P(x)N(x) = 2.79$$

bits/pixel compared to original 3bits/pixels. Theoretical optimum for single symbol encoding is the entropy $H_1 = 2.781$ bits/pixel.

Huffman Coding-Notes

- Obviously both coders and decoders must use same codebook.

- Codebooks either can be 'hardwired' into codecs as part of the standard (e.g Group 3 and 4 FAXm JPEG etc) (most common) OR

- Can transmit codebook first - but this reduces compression OR

- Can start with a default codebook and both transmitter and receiver can adapt using some algorithm a improved codebook as the symbols are decoded.

- Above basis of Lempel-Ziv-Welch (LZW) compression. Used in GIF and PNG image compression formats as well as unix *Compress*, *gzip* etc. For a signal with stationary statistics can show such universal compression approaches optimum lossless compression if signal of to H bits/pixel.

Multi-pixel Coding

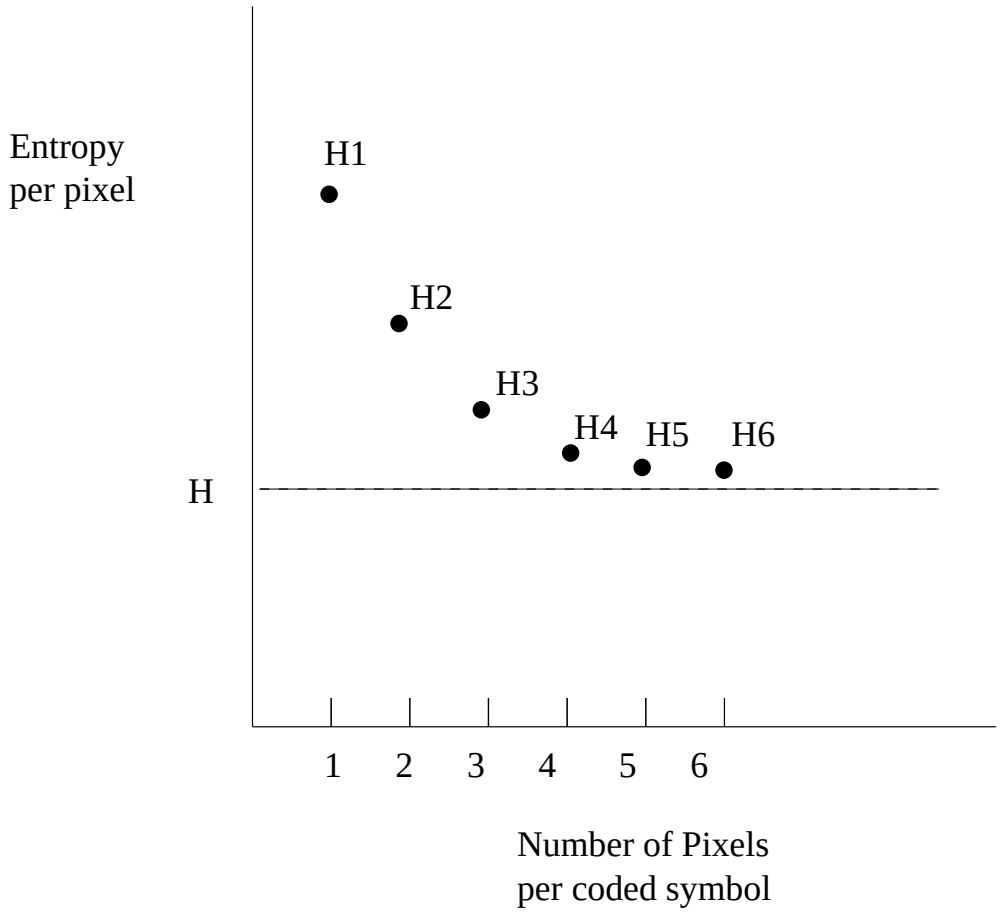
- In a 8bit image, each of the possible 256 graylevels has its own, variable length Huffman code, - data rate approaches single pixel entropy H_1.

- If adjacent pixels are correlated (as they always are in interesting images) can encode groups of 2 or more pixels, giving each a unique code. Exploits interpixel and coding redundancy

- Consider first encoding pairs of pixel values. There are now 256^2 possible 'symbols'. Based on probabilites of these 'symbols' can calculate a two pixel entropy of H_2 bits/symbol.

- If no correlation between pixels, no benefit, H_2 measured in bits/symbol is twice H_1 in bits/symbol. In terms of bits/pixel have same value and maximum lossless compression ratio same. However if pixels intercorrelated. H_2 in bits/pixel can be less than H_1 in bits/pixel, can increase compression.

- As the number of pixels jointly coded, n, goes to infinity we asymptotically obtain an estimate of the signal entropy $H = H_n$, which depends on the statistics of interpixel correlation. H bits/pixel is the theoretical limit for *any* type of lossless compression in a signal (Shannon).

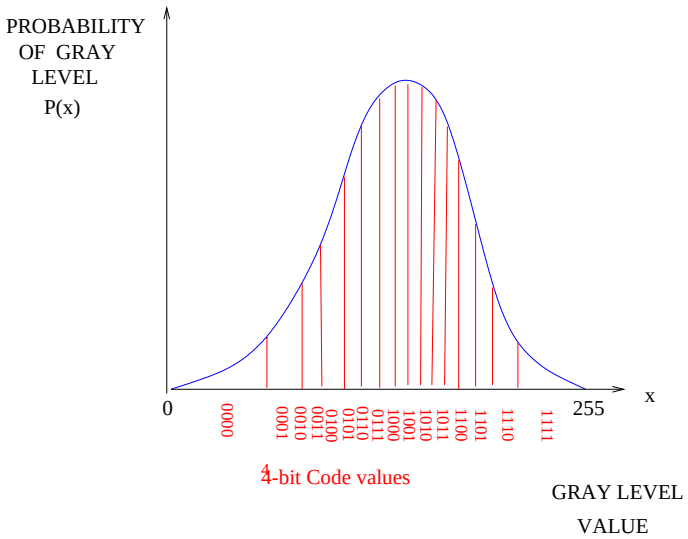


Quantiser

- Consider as a *lossy* alternative to entropy coding. Because it is lossy can achieve higher compression, but is not reversible(!)

- Go *from fixed length binary number* for each pixel value (say 8 bits) to another fixed length value (say 4 bits), and so compress by factor of 2.

- Simplest approach is just to remove 4 least significant bits from input value. Less distortion introduced if we convert pixels to 4-bit code words with unevenly distributed *decision levels*. Make decision levels closer together for regions where the image histogram peaks (p 368 of G & W).



- Given the pixel probabilities we have a requantisation table given below,

Gray levle Input Range (decimal)	Input Range (8bit binary)	Code value (4 bit Binary)	Reconstruction value
0 - 55	00000000 - 01100101	0000	27
56 - 70	01100110 - 10000110	0001	63
71 - 80	10000111 - 10100000	0010	76
.....			

etc, etc, etc

- We therefore have a compression of a factor of 2, although with a loss of image information. Choosing ranges for the requantisation based on the pixel histogram (narrow range at peak in histogram) however minimises errors.

- Theories exist on how for a given compression and image histogram to choose the decision levels and reconstruction values to minimise error.

Mapper

- We will discuss this more fully in the next lecture. Will consider both *waveform* and *transform* coders.

- Waveform encoders work in the the image domain. Simplest type, form difference between present pixel and the last one, then can quantise and entropy code this quanitity. Get more compression because the difference image has lower entropy.

- Will also discuss more complex predictive coding methods and run length coding.

- Transform coders use general transforms to another domain e.g. cosine or wavelet transforms used in JPEG and JPEG2000. Discuss JPEG in detail, since it uses psycho-visual redundancy, cosine transform, run-length coding and Huffman coding.

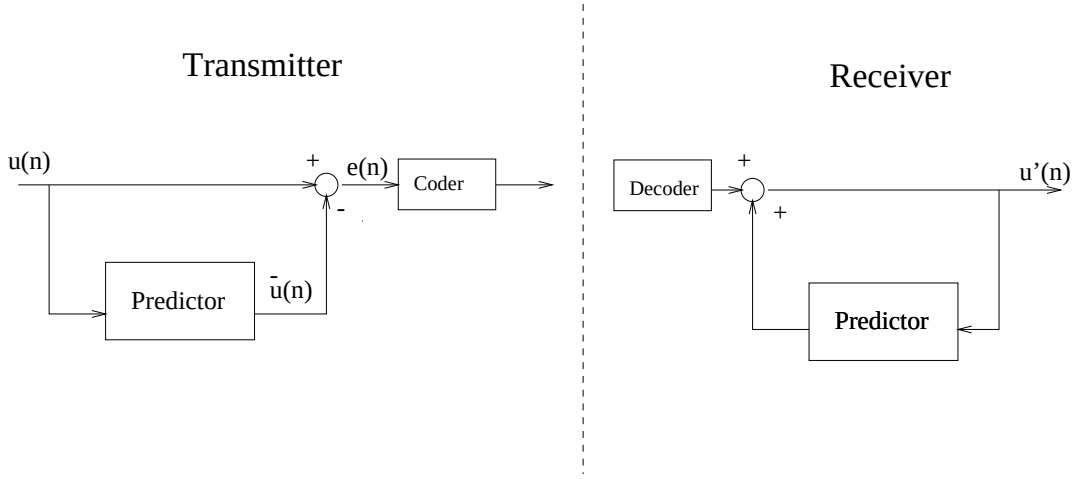
- In last lecture (if we have time) we will discuss briefly video compression (MPEG).

Simple Transformer - Pixel Difference Coding

- On each line of image form predictor of next pixel value based on previous pixels on a line $\bar{f}_n$ by some rule. The simplest predictor is $\bar{f}_n = f_{n-1}$, the last pixel value. Both transmitter and receiver use same rule.

- Transmit not f_n but the pixel error $e_n = f_n - \bar{f}_n$. If the predictor is simply the previous pixel values we encode *pixel differences*.

- Since pixels next to each other are usually highly correlated small differences are common. Histogram of differences is highly peaked and *low entropy*. Can often create an efficent entropy code for e_n, get good lossless compression.


Transmitter
Receiver
u(n)
+
e(n)
Coder
-
Predictor
ū(n)
Decoder
+
u'(n)
+
Predictor