7: Image Compression

Mark Handley

Image Compression

- GIF (Graphics Interchange Format)
- PNG (Portable Network Graphics)
- MNG (Multiple-image Network Graphics)
- JPEG (Join Picture Expert Group)

GIF (Graphics Interchange Format)

- Introduced by Compuserv in 1987 (GIF87a)
- Support for multiple images in one file and metadata adding in 1989 (GIF89a)
- Indexed image format: up to 256 colours per image, chosen from a variable palette.
  - One colour index can indicate transparency.
  - Uses lossless LZW compression of data bytes.
  - Optional interlacing capability.

LZW (Lempel-Ziv-Welch)

- LZW is a form of dictionary coding (based on LZ78).
  - Build a dictionary of words in the text to be encoded.
  - Send index into dictionary instead of word itself.
  - Example of dictionary encoding:

    Uncompressed text:

    ```
    "I am dumb and because I am dumb, I can't
     even tell you that I am dumb."
    ```

    Compressed text:

    ```
    "$1 and because $1, I can't even tell you
     that $1. $1=[I am dumb]"
    ```

LZW Compression

Dictionary starts with one entry for each possible byte value (256 entries).

```
STRING = get input character
WHILE there are still input characters {
  CHAR = get input character
  IF STRING+CHAR is in dictionary {
    STRING = STRING+CHAR
  } ELSE {
    output the code for STRING
    add STRING+CHAR to dictionary
    STRING = CHAR
  }
}
output the code for STRING
```

LZW Decompression

```
Read OLD_CODE
output OLD_CODE
WHILE there are still input characters {
  Read NEW_CODE
  STRING = get translation of NEW_CODE from dictionary
  output STRING
  CHAR = first character in STRING
  add OLD_CODE + CHAR to the dictionary
  OLD_CODE = NEW_CODE
}
```

- Nice property is that dictionary does not need to be sent - is rebuilt automatically at receiver.
- Actually slightly more complex than this - one exception.

GIF Uses

- GIF became very popular in the early days of the Web.
  - Supported by NCSA Mosaic.
  - Pretty good compression.
  - Most displays then were indexed rather than truecolor.
- Today it's still good for diagrams, cartoons, and other non-photographic images.
  - Lossless encoding good for sharp edges (doesn't blur).

GIF Patent Issues

- Compuserv designed GIF without knowing Unisys had a patent on LZW.
  - Long after LZW became popular, Unisys started to claim royalties on GIF implementations.
  - This prompted efforts to boycott GIF and spurred the development of PNG.
  - Original Unisys LZW patents now expired.

PNG (Portable Network Graphics)

- Supports truecolor, greyscale, and palette-based (8 bit) colourmaps.
- Uses DEFLATE algorithm:
  - As used in gzip
  - LZ77 algorithm with Huffman coding.
  - Patent free.
  - Spec: http://www.ietf.org/rfc/rfc1951.txt
- Combines this with prediction.
  - 5 different simple prediction algorithms can be used, chosen on a per-scanline basis.
  - Eg: sample-to-left, sample-above, average of s-t-l and s-a, etc.
  - DEFLATE only compresses the difference between the prediction and the actual value.

LZ77

- Unlike LZ78, uses the datastream as the dictionary.
- Keeps a history window of the recently seen data. Compares current data with history.
  - A match is encoded as:

    Length of match

    Position in history.
  - A non-match is encoded as a literal for “non-match” followed by the actual sample value.

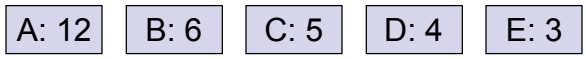
Huffman Coding

- Variable length coding, with most frequent codes using fewest bits and less frequent codes using more bits.
- Provably optimal.

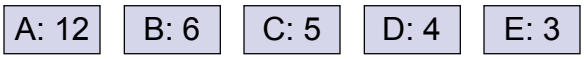
- Encoding done by building an encoding tree.
- Tree is built bottom up, based on frequencies of characters from the text.

Huffman Coding: Example

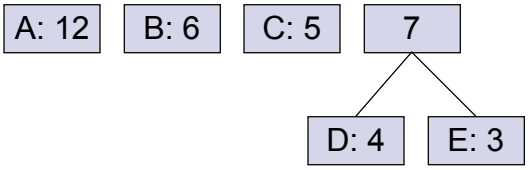
- Eg: five letter alphabet, A,B,C,D,E with frequencies in the text of A: 12, B: 6, C: 5, D: 4, E: 3
- Start with five separate subtrees:



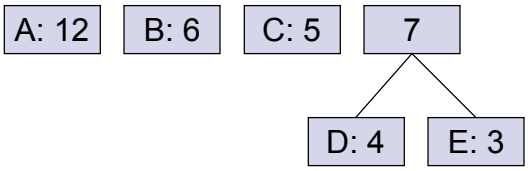
Huffman Coding: Example



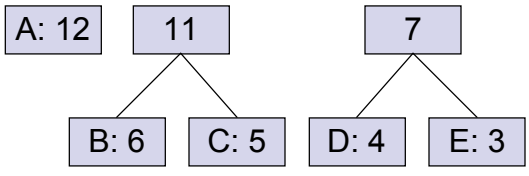
Subtrees for D and E have lowest frequencies, so merge them:



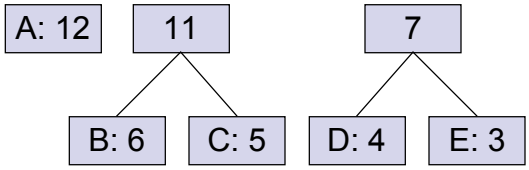
Huffman Coding: Example



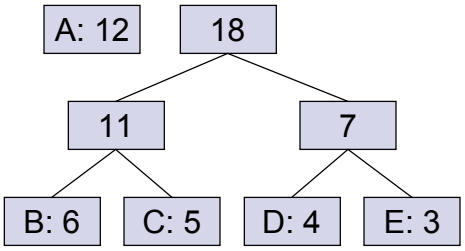
Subtrees for B and C have lowest frequencies, so merge them:



Huffman Coding: Example

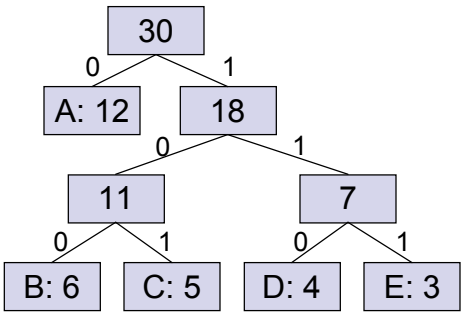


Subtrees for {B,C} and {D,E} have lowest frequencies; merge:



Huffman Coding: Example

Finally:



- To encode a character, walk down the tree. Each left branch adds a 0, each right branch adds a 1.
- Eg: A codes as 0, B codes as 100, E codes as 111

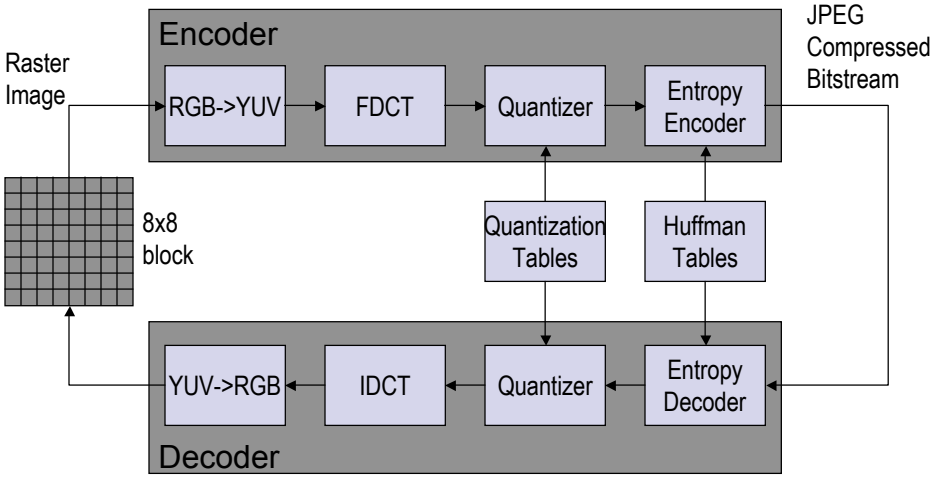
JPEG (Joint Photographic Experts Group)

- Good for compressing photographic images
  - Gradual changes in colour
- Not good for graphics
  - Sharp changes in colour.
- Compression ratio of 10:1 achievable without visible loss.
- Uses JFIF file format:
  - JPEG File Interchange Format
  - http://www.w3c.org/Graphics/JPEG/jfif3.pdf

JPEG

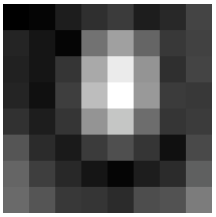
- Convert RGB (24 bit) data to YUV.
  - Typically YUV 4:2:0 used.
  - Three “sub-images”, one each for Y, U and V
  - U and V sub-images half the size in each dimension as Y
- Divide each image up into 8x8 tiles.
- Convert to frequency space using a two-dimensional DCT
- Quantize the frequency space, using more bits for the lower frequencies.
- Encode the quantized values using Run-length encoding and Huffman coding in a zig-zag manner.

JPEG Diagram



JPEG Example

Original 8x8 luminance block



- Actual values:

52	55	61	66	70	61	64	73
64	59	55	90	109	85	69	72
62	59	68	113	144	104	66	73
63	58	71	122	154	106	70	69
67	61	68	104	126	88	68	70
79	65	60	70	77	68	58	75
85	71	64	59	55	61	65	83
87	79	69	68	65	76	78	94

Subtract 128 from each value to convert to signed

Then apply FDCT:

$$T(i,j) = c_i c_j \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} V(y,x) cos \frac{(2y+1)i\pi}{2N} \cos \frac{(2x+1)j\pi}{2N}$$

$$c_i = \sqrt{1/N} \text{ if } i = 0, \; c_i = \sqrt{2/N} \text{ otherwise. Similarly } c_j$$

Giving:

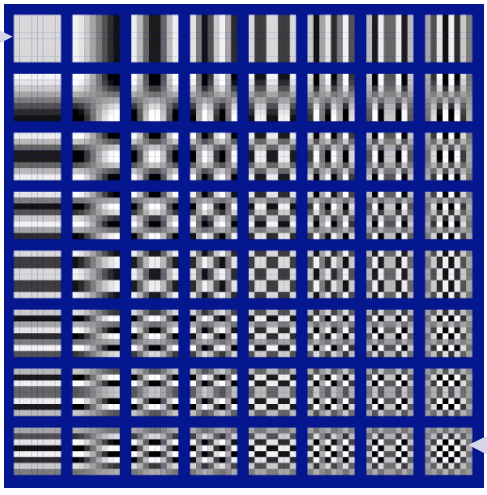
-415	-30	-61	27	56	-20	-2	0
5	-22	-61	10	13	-7	-8	5
-47	7	77	-24	-29	10	5	-6
-49	12	34	-15	-10	6	2	2
12	-7	-13	-4	-2	2	-3	3
-8	3	2	-6	-3	1	4	2
-1	0	0	-3	-1	-3	4	-1
0	0	-1	-4	-1	0	0	2

Note DC Coefficient has lots of power

Very little power in high frequencies

DCT Basis Functions

[what each coefficient corresponds to in the image]



Quantize using a quantization matrix such as:

16	11	10	16	24	40	51	61
12	12	14	19	26	58	60	55
14	13	16	24	40	57	69	56
14	17	22	29	51	87	80	62
18	22	37	56	68	109	103	77
24	35	55	64	81	104	113	92
49	64	78	87	103	121	120	101
72	92	95	98	112	100	103	99

Better quantization at low frequencies

Coarse quantization at high frequencies

Eg round(-415/16) = -26

Giving:

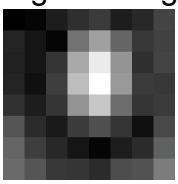
-26	-3	-6	2	2	-1	0	0
0	-2	-4	1	1	0	0	0
-3	1	5	-1	-1	0	0	0
-4	1	2	-1	0	0	0	0
1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

High frequencies often quantize to zero

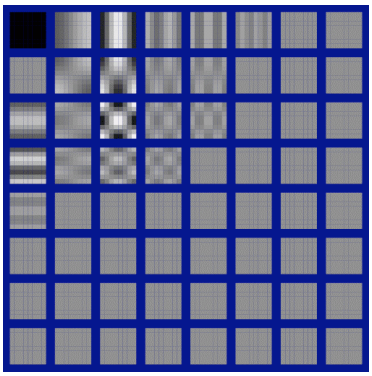
Quantized DCT coefficients:

-26	-3	-6	2	2	-1	0	0
0	-2	-4	1	1	0	0	0
-3	1	5	-1	-1	0	0	0
-4	1	2	-1	0	0	0	0
1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Original Image:

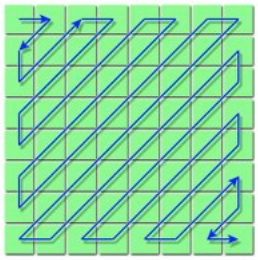


Scaled DCT basis functions that make up the (quantized) image



Order the coefficients in zig-zag order:

$$\begin{pmatrix} -26 & -3 & -6 & 2 & 2 & -1 & 0 & 0 \\ 0 & -2 & -4 & 1 & 1 & 0 & 0 & 0 \\ -3 & 1 & 5 & -1 & -1 & 0 & 0 & 0 \\ -4 & 1 & 2 & -1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$



−26, −3, 0, −3, −2, −6, 2, −4, 1, −4, 1, 1, 5, 1, 2, −1, 1, −1, 2, 0, 0, 0, 0, 0, −1, −1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,

Run-length encode:

−26, −3, 0, −3, −2, −6, 2, −4, 1, −4, {2 x 1}, 5, 1, 2, −1, 1, −1, 2, {5 x 0} , −1, −1, EOB

Huffman code what remains. Encoding is complete.

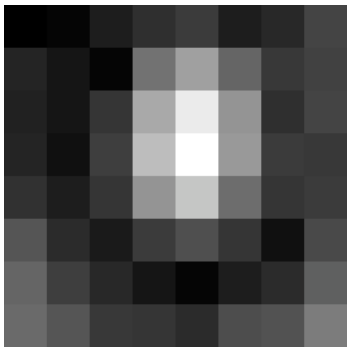
JPEG Decoding

- Decoding is simply the reverse of encoding.
- Reverse the huffman, RLE encodings.
- Dequantize.
- Apply inverse DCT (IDCT):

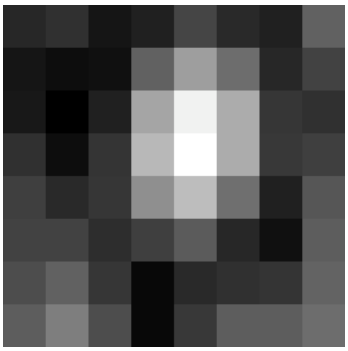
$$V(x,y) = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} c_i c_j T(i,j) cos \frac{(2y+1)i\pi}{2N} \cos \frac{(2x+1)j\pi}{2N}$$

- Add 128 to convert back to unsigned.

Comparison



Original Image



Decompressed Image

JPEG Compression ratio

- Compression ratio depends on how large the values in the quantization matrix are.
- 10:1 achievable without noticeable loss.
- 100:1 achievable, but artifacts are noticeable.