Image Compression

1 Redundancy in images

We can compress images by targeting three principal types of redundancy

Definition: Coding redundancy

Use of sub-optimal codes means that we may use more bits than are needed to represent the pixel values

Definition: Spatial redundancy

Neighbouring pixels are likely to have similar values. Information may be unnecessarily replicated in the representation of spatially correlated pixels

Definition: Irrelevant information

Images may contain visually non essential information that is ignored by the human visual system

2 Lossy vs Lossless Compression

Lossy compression:

- Used when images need not be reproduced exactly and an approximation is OK
- Compression artefacts in the image, which may not be visible
- Source of noise for image processing and computer vision algorithms
- Widely used, efficient implementations exists

Lossless compression:

- Computationally more expensive
- Resulting file size often larger than corresponding lossy compression files
- No additional noise to the image
- Generally less widely used

3 JPEG

JPEG is an image compression algorithm based on the Discrete Cosine Transform and variable length encoding.

Offers tunable (user controlled quality) lossy compression

3.1 DCT

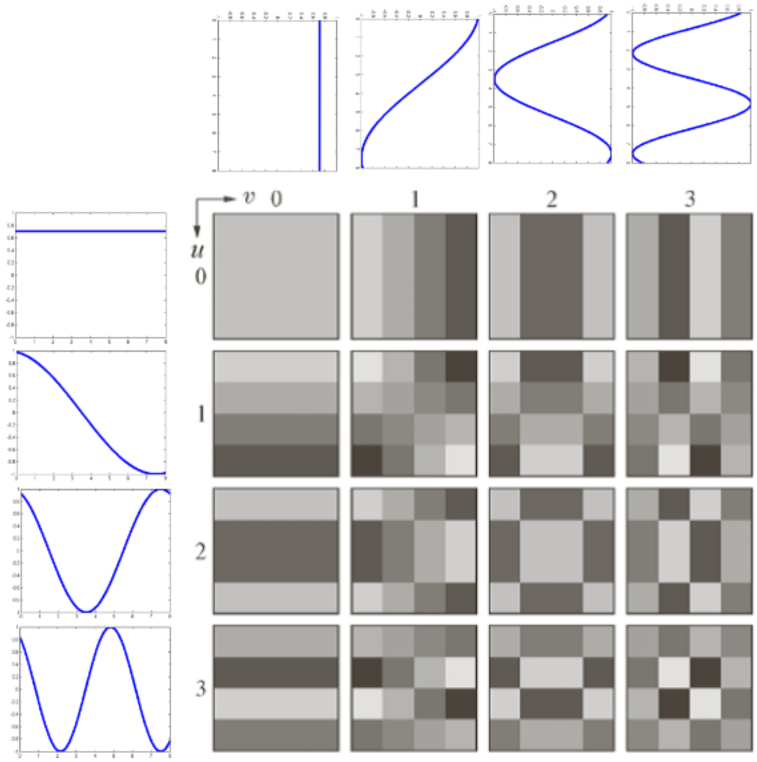
The 1D DCT expresses a vector of data points, here pixel intensity values, as a weighted sum of cosine functions of varying frequencies. Essentially a variation of DFT.

Similarly to the DFT, DCT can also be seen as a mathematical change of basis, which can be described as multiplication of the data vector by a special matrix, here the DCT matrix.

Unlike the DFT, DCT is a real transform and not a complex one. The elements of the DCT matrix and real numbers.

3.1.1 Basis functions

The 2D DCT uses a set of 2D matrices as basis functions, each one corresponding to a 2D cosine function



3.2 JPEG Compression

Step 1 - Subdivide the image into pixel blocks of 8x8 size. The blocks are processed one after the other from left to right and top to bottom.

Step 2 - Assuming that the values of the pixels are in the range [0,255], subtract 128 to bring them into the range [-128,127]. This is done as the DCT maps that interval on itself.

Step 3 - Apply the 8x8 DCT to each block. The DCT values are computed with 11-bit precision (even though the input has 8-bit precision)

Step 4 - Scale and quantize the DCT values, using the following quantisation matrix

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

That is, if $T(u,v)$ is the DCT of the 8x8 pixel block, we do component wise division of the two matrices and round the result

$$\hat{T}(u,v) = \text{round}\left[\frac{T(u,v)}{Z(u,v)}\right]$$

The quality of the image is controlled by a user parameter which determines the Z matrix

Step 5 - create a sequence of the quantised DCT coefficients $\hat{T}(u, v)$ using the zig zag pattern

0	1	5	6	14	15	27	28
2	4	7	13	16	26	29	42
3	8	12	17	25	30	41	43
9	11	18	24	31	40	44	53
10	19	23	32	39	45	52	54
20	22	33	38	46	51	55	60
21	34	37	47	50	56	59	61
35	36	48	49	57	58	62	63

Step 6 - Encode the sequence of quantised coefficients $\hat{T}(u, v)$ obtained in step 5 with a Huffman based variable length code, encoding each coefficients value and the number of preceeding zeros.

Use a special symbol for the end of non-zero coefficients