IMAGE Report 3 - Image Compression

Carl Robinson & Majd Abazid **17th Nov 2017**

1) - Quality Control and Quantisation

JPEG is a lossy compression format that provides some of the best compression factors on the market (20:1 to 25:1 without noticeable loss of quality). JPEG is much more effective on photographic images (containing many different coloured pixels, gentle colour gradients, and noise) than it is with geometrics/graphics images (which contain large blocks of solid colour separated by sharp boundaries). Especially in the latter case, JPEG compression can introduce visible artefacts.

A summary of the JPEG algorithm is below. The information loss (and the consequent introduction of artefacts) occurs during the quantisation stage (quantification in french).

Quality control in image software is related to the quantisation stage of the image processing chain. The quality control of JPEG compression often ranges from 1 to 100 (GIMP) or from 1 to 12 (Photoshop). Specifically it controls the size of the quantisation step, such that a low JPEG quality setting is related to a large quantisation step, and a high JPEG quality setting is related to a small quantisation step.

In the first stage of the image processing chain, the image is transformed into discrete cosine coefficients, in order to separate the low and high frequencies. In the second stage, these coefficients are quantised, causing some information loss. Large quantisation steps reduce a larger range of DCT coefficients down to a single value, thereby improving compression by reducing the number of unique DCT values to code in the following step. However, the larger the quantisation step is, the more information is lost in the quantisation process, which causes a greater reduction in image quality.

Additionally, the quantisation step size is not applied equally to all DCT coefficients. The DCT coefficients responsible for representing the high frequencies are subject to a larger quantisation step than the low frequencies, whatever the quality control setting chosen. This is the case because the human eye is less sensitive to high frequencies (the details) in images, and more sensitive to low frequencies (the large objects). By deliberately removing more of the high frequencies from the image, its size can be reduced without any noticeable difference to the human eye.

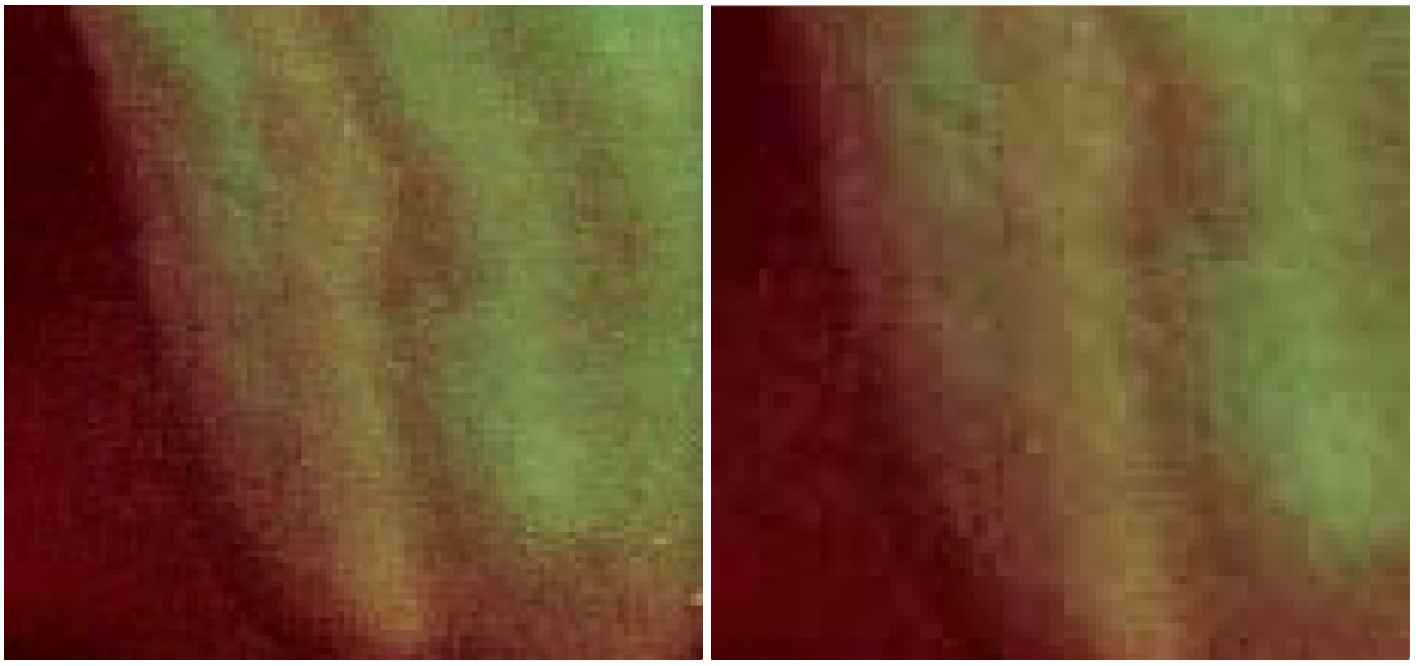
2) - Quality and BPP

Peppers photo

	Non-Optimised		Optimised	
Quality	Size in bits	Bits per pixel	Size in bits	Bits per pixel
2	400920	1.529	396256	1.512
4	493408	1.882	439104	1.675
6	670472	2.558	607904	2.319
8	797760	3.043	780816	2.979
10	1149728	4.386	1100080	4.196
12	1921064	7.328	1695712	6.469

- Compression ratio from quality 12 to quality 2 = 400920 / 1921064 = 0.207

Peppers, non-optimised, quality level 6 (left) and quality level 4 (right)



- No Blocks at quality 6 (bpp 2.558) . Blocks begin to appear at quality 4 (bpp 1.882). This distortion is due to the quantification of the DCT coefficients, which have been modified to a different degree in each 8*8 pixel block, causing a noticeable difference on the boundary of each block.

Peppers, non-optimised, quality level 4 (left) and quality level 2 (right)

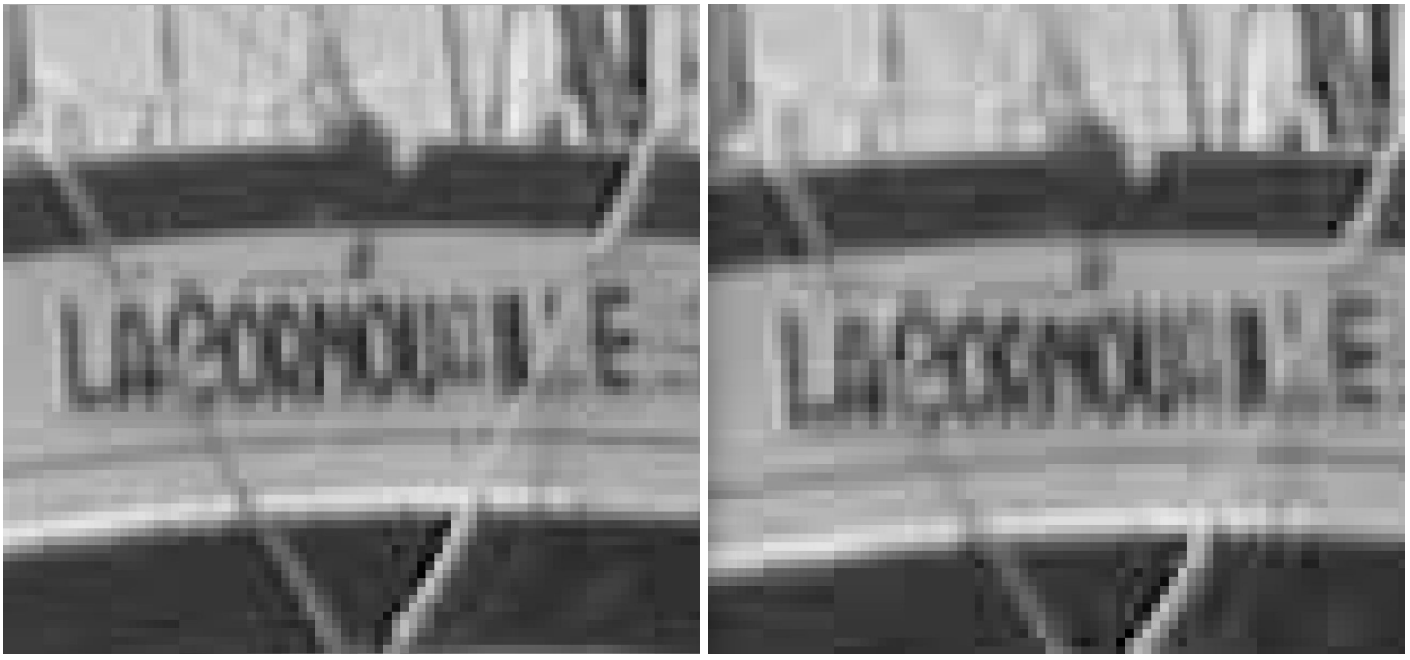
- Edges appear mostly fine at quality 4 (bpp 1.882). They start to degrade and create a halo effect on the red background at quality 2 (bpp 1.529).
- Degradation of edges occurs because the high frequencies, which are responsible for the details in the image, have been reduced. Therefore the compressed image is dominated by the average value of each 8*8 pixel block. In this section of the same image there are fewer gentle variations in colour, and more sharp changes (hard edges). Therefore the degradation of edges is more visible.
- The blockiness appears at a higher quality level than the degraded edges because the blockiness occurs mainly on the large objects and across surfaces of gentle gradients, and the human eye is more sensitive to changes in large objects / low-frequencies.

Lacornou photo

	Non-Optimised		Optimised	
Quality	Size in bits	Bits per pixel	Size in bits	Bits per pixel
2	281672	4.298	279504	4.265
4	309672	4.725	307920	4.698
6	291992	4.455	290560	4.434
8	359232	5.481	357544	5.456
10	438560	6.692	381504	5.821
12	571264	8.717	517912	7.903

- Compression ratio from quality 12 to quality 2 = 281672 / 571264 = 0.493
- This compression ratio is much worse (much higher) than the other two images.

Lacornou, non-optimised, quality level 4 (left) and quality level 2 (right)



- The edges look OK at quality 4, and there is no visible blockiness. As seen from the text on the boat, the edges begin to degrade at quality 2 (bpp 4.298) and blockiness appears too. These occur at the same quality level as there are fewer areas of slow change in the image, which is usually where blockiness is found first.
- Oddly the file size is smaller (and therefore the bpp is lower) at quality level 6 than quality level 4. This could be due to the specific frequency distribution in this image, such that the quantisation step for quality level 6 resulted in fewer unique DCT coefficients, which allowed for more effective entropy coding in the next stage of the processing chain.

Logo graphic

	Non-Optimised		Optimised	
Quality	Size in bits	Bits per pixel	Size in bits	Bits per pixel
2	483776	1.385	442256	1.266
4	499936	1.431	510960	1.463
6	573816	1.643	583352	1.670
8	719264	2.059	610856	1.749
10	958952	2.746	817240	2.340
12	2208456	6.323	1779832	5.096

- Compression ratio from quality 12 to quality 2 = 483776 / 2208456 = 0.219
- This is surprisingly good considering that JPEG is not designed to compress this type of graphics image. However, the quality of the result is less impressive, as described below.

Logo, non-optimised, quality level 12 (left) and quality level 2 (right)

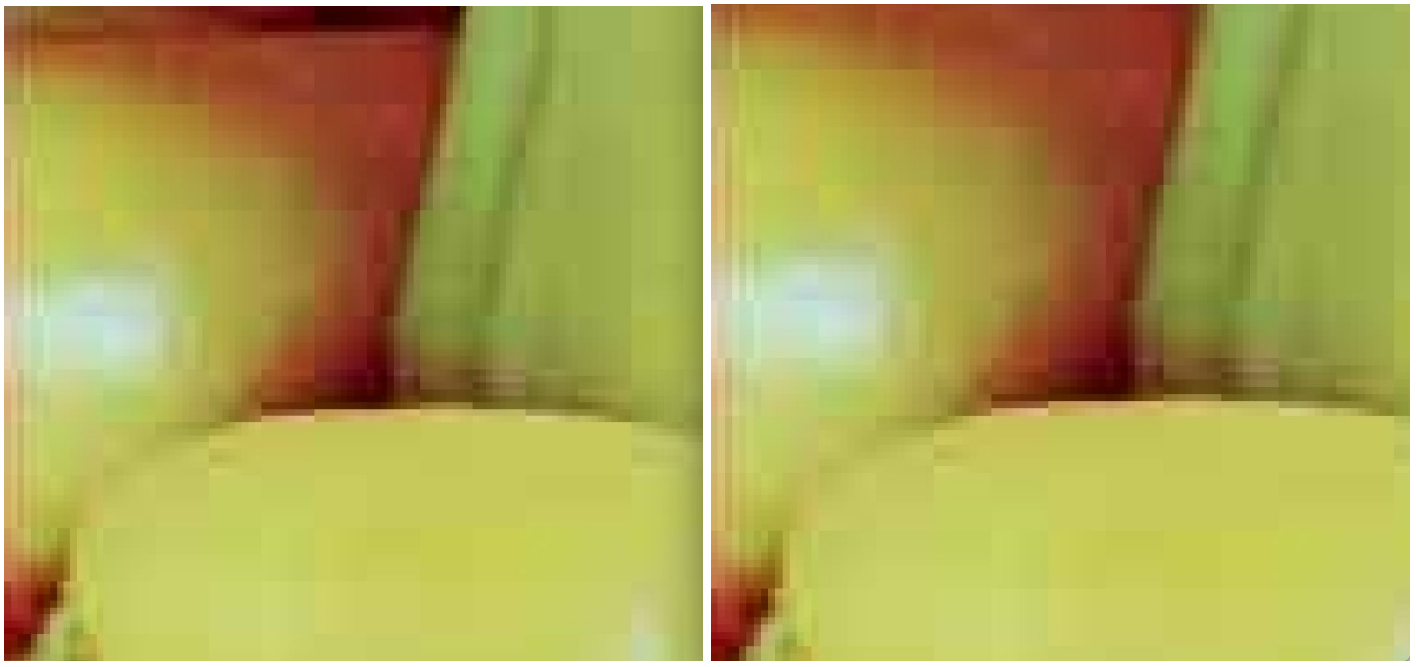
- Edges start to bleed into the areas of solid colour even at the highest quality level 12 (bpp 6.323). At lower levels, such as quality level 2 (bpp 1.385), the effect is much more noticeable. This is because the sharp changes in colour at the edges are represented by very high frequencies, which are the first to be attenuated by the JPEG compression. In the quantisation step, the mix is set to attenuate high frequencies more than low frequencies, as the eye is less sensitive to details in photographs of natural phenomenon. However, in logo images such this one, this process causes undesirable artefacts to develop around the edges of objects.
- The blockiness only appears at the lowest levels (quality 2), and is hard to see. This is due to the complete lack of colour gradients in the image. The blockiness occurs only at the boundaries of two colours. Otherwise the first DCT component is used to represent the areas of solid colour.

Conclusions

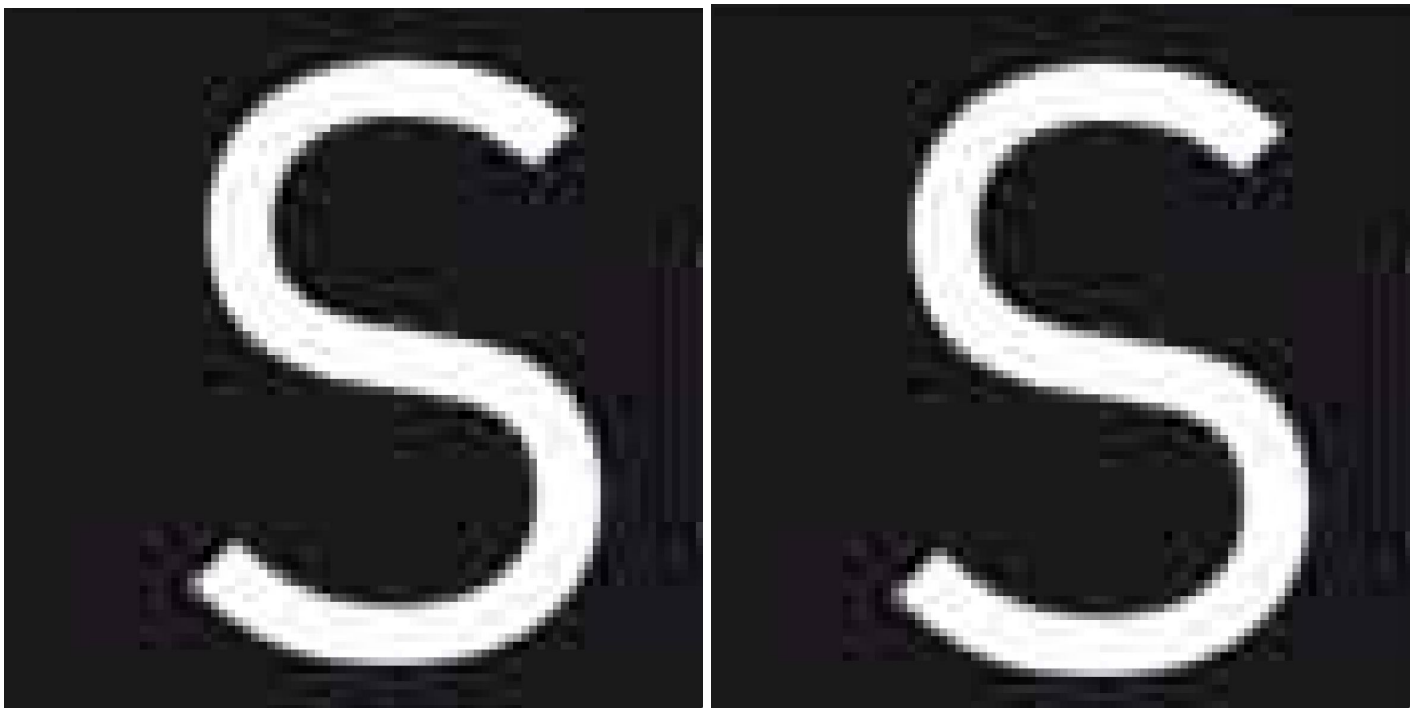
- Reducing quality means increasing size of the quantisation step, which leads to the loss of information responsible for fine details in the image. Fortunately this also includes noise, which is removed.
- For the areas of the original peppers image that change quickly (high frequencies), such as the change from highlight to shadow, we see a blocky effect in the compressed image. On the other hand, we see areas of solid colour where there was more gradual change in the original image.
- There is little degradation of the edges on the peppers, as the contours are already blurred (slow frequencies) in the original image, so they are represented by lower frequencies which are not attenuated as much by the DCT transformation.
- The compression of the logo is much worse than compression of the two photos because the sharp discontinuity of values between adjacent blocks of solid colour require small quantification steps for high frequencies. By default, JPEG assigns a larger quantification step to high frequencies, and introduces oscillations around these discontinuities. The compression rate is therefore worse, and the compressed image is of visibly poorer quality as a result.

3) - Entropy Coding

Peppers, quality level 2, non-optimised (left) and optimised (right)



Logo, quality level 2, non-optimised (left) and optimised (right)



- In both cases we get exactly the same image after optimisation. Even the artefacts are the same.

Peppers photo

- Optimisation ratio for quality 12 = 1695712 / 1921064 = 0.883
- Optimisation ratio for quality 2 = 396256 / 400920 = 0.988
- At high quality, the optimised file is **88.3%** the size of the non-optimised one, and at low quality, the optimised file is **98.8%** the size of the non-optimised one i.e. optimisation hardly made any difference at all to the low quality file size. This is because the low quality file has lower entropy than the high quality file, as the DCT coefficients have been quantised to a greater degree (with larger steps). Huffman coding in particular is less effective on low entropy data, as there are much fewer values with which to build the tree, and so the tree has fewer levels, diminishing the optimisation benefit afforded by this structure.

Lacornou photo

- Optimisation ratio for quality 12 = 517912 / 571264 = 0.907
- Optimisation ratio for quality 2 = 279504 / 281672 = 0.992
- At high quality, the optimised file is **90.7%** the size of the non-optimised one, and at low quality, the optimised file is **99.2%** the size of the non-optimised one i.e. optimisation hardly made any difference at all to the file size.
- The high quality Lacornou image is very detailed, meaning that many high frequencies are still present after the quantisation stage, so the resulting DCT coefficients have high entropy. This greatly reduces the effectiveness of the entropy coding.
- RLE encoding: All the DCT coefficients of each block are coded with RLE, except the continuous component which is coded by differential coding relative to the previous block. A high entropy input means that many DCT coefficients will be involved and each will have a low probability. Each of the ‘words’ that encode them will therefore be long, and so the average number of bits per word (or in this case bits per pixel, BPP), will be high. The efficiency of run length encoding (RLE) depends largely on the characteristics of the image. This image has many fine details, which means the pixel values often change from one pixel to the next. RLE is not designed to optimise sequences which change rapidly, so the optimisation is less effective.
- Huffman encoding: Having many infrequently occurring (low-probability) DCT coefficients leads to the creation of a deep Huffman tree, and therefore long encoding words for each DCT coefficient. Long words increase the BPP, reducing the effectiveness of compression.

Logo graphic

- Optimisation ratio for quality 12 = 1779832 / 2208456 = 0.806
- Optimisation ratio for quality 2 = 442256 / 483776 = 0.914
- For the logo, at high quality, the optimised file is **80.6%** the size of the non-optimised one, and at low quality the optimised file is **91.4%** the size of the non-optimised one.
- The optimisation worked much better for the logo than it did for the two photos due to the fact there are very few colour variations in the image. After RLE encoding, only a low number of bits is needed to code each pixel.

General points

- Entropy coding is reversible and lossless (the only stage that causes loss of information is the quantification stage).
- Coding of the 64 DCT coefficients from each 8*8 pixel block is as follows:
    - For the continuous component we calculate the difference between the values of the previous blocks (differential coding)
    - For the other 63 coefficients we code them using RLE.
    - The 64 values are then coded using an entropy coding such as Huffman.
    - None of these stages causes loss of information.

4) GIF vs JPEG - BPP Comparison

	Size in bits	Bits per pixel
lacornou.gif	545112	8.318
lacornou.pgm	524408	8.002
logoTSP.gif	406240	1.163
logoTSP.ppm	8383048	24.001
peppers.gif	1190424	4.541
peppers.ppm	6291760	24.001

- The original peppers.ppm and logoTSP.ppm images are 24 bit bitmap images, so of course they have 24 bits per pixel (they use 8 bits for each of red, green and blue component colours). The original lacornou.pgm is an 8 bit greyscale bitmap image that uses 8 bits to represent 256 shades of grey, and therefore uses 8 bits per pixel.

Peppers photo

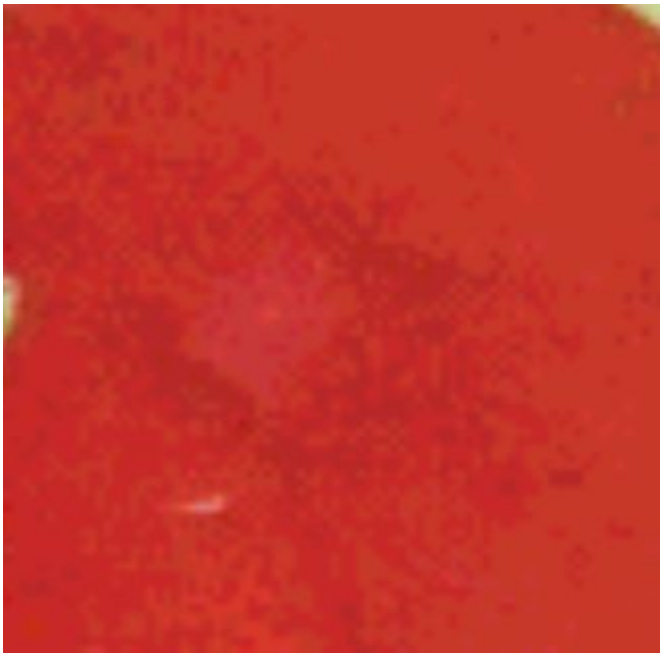
GIF

- The GIF compression reduced the peppers.ppm bitmap from 24 bpp to 4.541 bpp, with a respectable compression down to 18.92% of the original file size.
- If the image has more than 256 colours, as it does here, GIF will reduce this down to 256 only. As seen in the example image, this can cause degradations in the colour gradients, where areas of solid colour replace subtle gradients.

JPEG - the most efficient

- Edges appear mostly fine at quality 4 (bpp 1.882). They start to degrade and create a halo effect on the red background at quality 2 (bpp 1.529). This is a much lower bpp than the gif image for a better quality image.

JPEG



GIF

This is why we prefer to use **JPEG** for compressing this image.

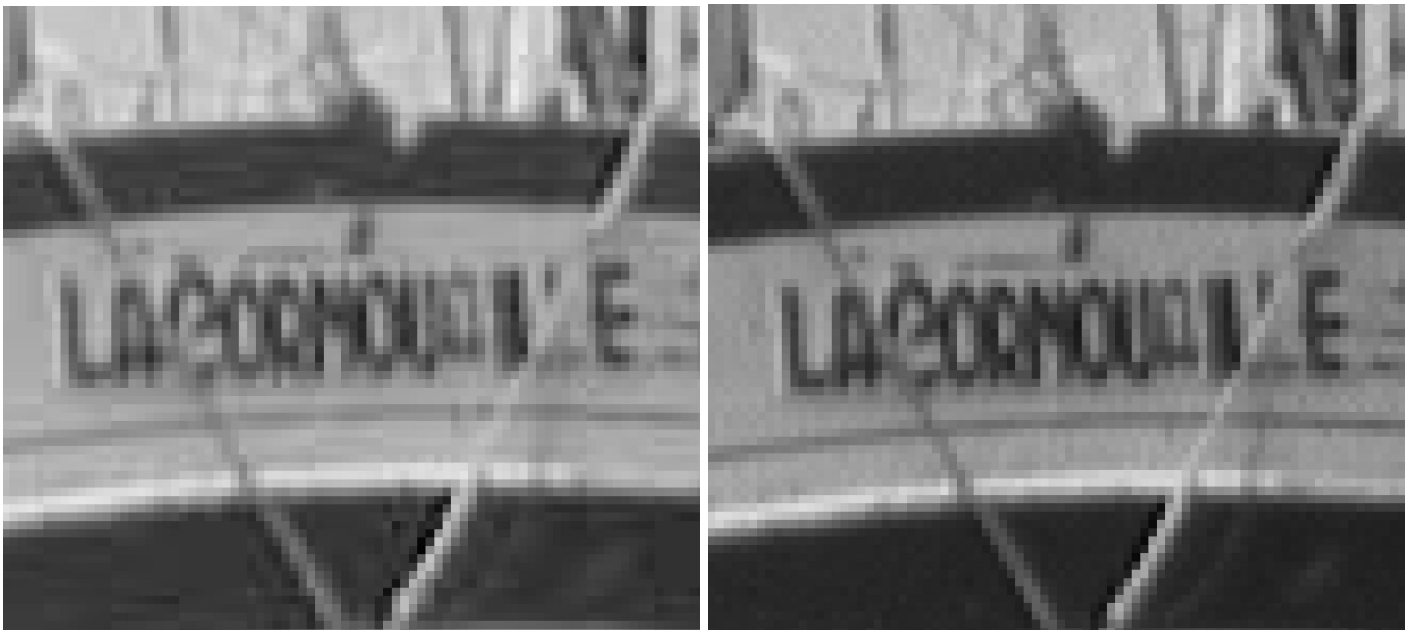
Lacornou photo

GIF

- The GIF compression actually *increased* the size of the lacornou.pgm image, increasing the bpp from 8.001 to 8.318. As mentioned before, this is because the GIF coding algorithm is optimised for graphics images, not photos.
- GIF is reversible for images that contain 256 colours or less, so in these cases it is not a lossy compression format.

JPEG - the most efficient

- The JPEG compression reduced the lacornou.ppm bitmap from 8.313 bpp to 4.298 bpp at quality 2, which is a mediocre compression of 51.59% of the original file size.
- The edges only begin to degrade at quality 2 (bpp 4.298), so compared to GIF, a better level of compression could be attained before degradation became apparent. The degradation of the edges is visible, but we can still see the fine details in the image.



JPEG GIF

This is why we prefer to use **JPEG** for compressing this image.

Logo graphic

GIF - the most efficient

- The GIF compression reduced the logoTSP.ppm bitmap from 24 bpp to only 1.163 bpp. The compressed image is only 4.85% the size of the original, which is an excellent compression ratio.
- The GIF compressed image has very little degradation at all, and is an excellent quality result considering the compression ratio.
- Furthermore, the image has fewer than 256 colours, so it is a lossless compression.

JPEG

- In contrast, the JPEG compression produces a lot of degradation and artefacts, as it introduces oscillations around the colour boundaries. These edges start to bleed into the adjacent areas of solid colour even at the highest quality level 12 (bpp 6.323), which is a much higher bpp than that achieved with GIF compression.
- Furthermore, the compressed image is 26.3% the size of the original, which is a mediocre compression ratio.

JPEG

GIF

This is why we prefer to use **GIF** for compressing this image.

5) GIF vs JPEG - Algorithm Comparison

JPEG

- JPEG allows us to compress images with loss, while controlling the level of visible degradation. JPEG uses discrete cosine transformation (DCT) to decompose 8*8 blocks of the image over sinusoidal functions into DCT coefficients.
- It compresses the photographic images pepper.ppm and lacornou.pgm well, but graphics such as logoTSP.ppm poorly
- RLE encoding: All the DCT coefficients of each block are then coded with RLE, except the continuous component which is coded by differential coding relative to the previous block. A high entropy input (e.g. lacornou) means that many DCT coefficients will be involved and each will have a low probability. Each of the 'words' that encode them will therefore be long, and so the average number of bits per word (or in this case bits per pixel, BPP), will be high. The efficiency of run length encoding (RLE) depends largely on the characteristics of the image. The lacornou image has many fine details, which means the pixel values often change from one pixel to the next. RLE is not designed to optimise sequences which change rapidly, so the optimisation is less effective.
- Huffman encoding: Having many infrequently occurring (low-probability) DCT coefficients leads to the creation of a deep Huffman tree, and therefore long encoding words for each DCT coefficient. Long words increase the BPP, reducing the effectiveness of compression. This is the case for lacornou.pgm, but not so for pepper.ppm, which has large areas of more gentle colour gradients corresponding to low frequencies. For pepper.ppm, most of the DCT coefficients will be those corresponding to the low frequencies, which will be coded with shorter words.

GIF

- GIF uses the LZW algorithm to encode chains of pixel values into a dictionary of coded values specific to that image. As such it efficiently compresses areas of uniform colour, as seen in the logoTSP.ppm image, as small codes (low number of bits) can be used to represent the most frequent colour values that appear in the image.
- It also limits the number of possible colour values to 256, so while information can be lost during compression, the file size is further reduced. If the original image has less than 256 colours, the GIF compression is reversible.
- GIF is the method of choice for graphical images such as web buttons, diagrams and logos, such as the logoTSP.ppm.
- Because of frequent small variations in consecutive pixels, photographs contain very few chains of repeated pixel values. As GIF relies on LZW encoding only, it is not very effective at compressing this type of image. GIF is also sensitive to noise as any noise causes new entries to be added to the dictionary, which decreases the efficiency of the compression. Complex, noisy photographic images such as lacornou.pgm are not suited to GIF compression, which is why it failed to compress this image.