2.2.2 An example: JPEG

Do the math, each pixel of a standard color image requires 3 bytes, then a 3648×2736 photo (this is the size provided by my low quality old digital camera) requires something like 30 megas. Why in your computer or in your cell phone do they need much less space? The answer is that they are compressed but not as something like you get when you apply `gzip` to a text file containing your assignment or your essay for a course that (fortunately) can be recovered without changing a comma. The kind of compression commonly applied to photos is lossy compression, you lose some information in an irreversible way. The camera and other devices know what information to lose keeping the visual aspect thanks to an algorithm heavily based on discrete Fourier analysis.

To say the whole truth, professional cameras allow to use the RAW image format that, as the name indicates, contains the signal almost as it comes out from the sensor. For mortals and even for arguably the most of the professional applications, the king format is JPEG also known as JPG. The name, different from the official original one, is actually metonymical because these letters stands for *Joint Photographic Experts Group*, the team that introduced the specifications of the format.

The treatment of color does not add too much to the essence of the method and involves some extra technicalities, then we worry firstly about B/W images (meaning colored with shades of gray). Our camera or scanner have already performed sampling and quantization and the image (signal) in a digital device is an integer matrix $C = (c_{jk})_{j=1,\,k=1}^{H\quad W}$ where $c_{jk} \in \{0, 1, 2, \ldots, 255\}$ specifies the gray tone of the pixel (j, k), being 0 black and 255 white. Here H is the height and W is the width, in pixels.

A first step is the subdivision of the image surrogate C into blocks of size 8×8. In some sense JPEG process individually these tiny blocks taking each few millimeters on your screen but it is important for the final compression that many of these blocks give alike outputs after signal processing. A problem appears at the edges if $8 \nmid H$ or $8 \nmid L$. In this case one invents artificially new pixels at the border to force H and L to be multiples of 8. We disregard this point here.

Each block can be considered as a function

$$F : \mathbb{Z}/8\mathbb{Z} \times \mathbb{Z}/8\mathbb{Z} \longrightarrow \{0, 1, 2, \ldots, 255\} \subset \mathbb{R} \tag{2.70}$$

where the arguments of F indicate the position in the block matrix B i.e., $F(\overline{j}, \overline{k}) = b_{jk}$ for $j, k \in \{0, 1, \ldots, 7\}$ where here and hereafter in this section we consider matrix indexes starting from 0.

We can write the inversion formula of Proposition 2.2.2 as

$$x_m = \frac{1}{N}\sum_{n=0}^{N-1} \alpha_n \widehat{x}_n^c \varphi_n(m) \quad \text{with} \quad \varphi_n(m) = \cos\left(\frac{\pi n}{N}(m + \frac{1}{2})\right) \quad \text{and} \quad \alpha_n = \begin{cases} 2 & \text{si } n \neq 0, \\ 1 & \text{si } n = 0. \end{cases} \tag{2.71}$$

If we apply it with $N = 8$ to $F(\cdot, l)$ and to $F(k, \cdot)$ i.e., taking discrete cosine transforms on

each variable, we have the discrete Fourier expansion

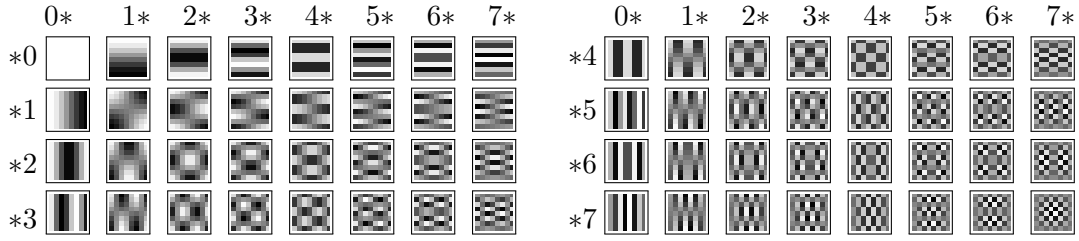
$$F = \sum_{n=0}^{7}\sum_{m=0}^{7} a_{nm}\phi_{nm} \quad \text{with} \quad \phi_{nm}(k,l) = \varphi_n(k)\varphi_m(l) \tag{2.72}$$

and

$$a_{nm} = \frac{\alpha_n\alpha_m}{64}\sum_{k=0}^{7}\sum_{l=0}^{7} F(k,l)\phi_{nm}(k,l). \tag{2.73}$$

Hence each coefficient requires a sum of 64 values and the computation by brute force of $\{a_{nm}\}_{n,m=0}^{7}$ requires roughly $64^2 = 4096$ operations per block. We have $W/8 \times H/8$ blocks and the resulting number of operations for a photo is affordable (see the comments at the end of this section).

We can read (2.72) saying that any 8×8 image represented by F can be written as a superposition of harmonics ϕ_{nm}. If we assign conventionally -1 to black and 1 to white, these harmonics can be seen as the following fundamental images:



For smooth blocks, Lemma 2.2.1 (recall that DCT is DFT forcing periodicity) implies that a_{nm} is small for the higher values of n and m. A usual photo contains many gradients, then for most blocks we could have a good reconstruction using only the a_{nm}'s with small index. On the other hand, it is clear that from a distance it is easier to distinguish an internal structure in the stripes of ϕ_{04} than in the checkerboard of ϕ_{77}. There are also effects related to the neurology of vision. This suggests to use a nonuniform quantizer giving few levels in general for higher values of n and m. In practice, one fixes a *quantization matrix* $Q \in \mathcal{M}_{8\times 8}(\mathbb{Z})$ and applies the uniform quantizer (2.13) with $\Delta = q_{nm}$. In other words, the quantization is

$$a_{nm} \longmapsto q_{nm}\widetilde{e}_{nm} \qquad \text{with} \quad \widetilde{e}_{nm} = \lfloor q_{nm}^{-1}a_{nm} + 1/2 \rfloor. \tag{2.74}$$

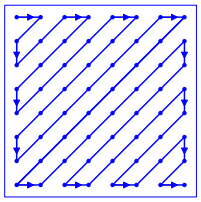
In this way, smaller values of q_{nm} give less spaced levels and hence a finer approximation to a_{nm}. The quantization matrix is not forced by the standards of JPEG format. In principle one could change it manipulating the header of the file but in reality everybody (every software) blindly uses the recommended quantization matrix

$$Q = \begin{pmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{pmatrix}. \tag{2.75}$$

According to the official documentation [CCI91, Annex K] this and another quantization matrix that we shall see later "are based on psychovisual thresholding and are derived empirically [...]. These tables are provided as examples only and are not necessarily suitable for any particular application" and "If these quantization values are divided by 2, the resulting reconstructed image is usually nearly indistinguishable from the source image". As expected, the northwest triangle contains smaller numbers than the southeast triangle because the former corresponds to more visible frequencies. A curious property is that Q is only approximately symmetric. Probably it reflects a minor asymmetry in our vision.

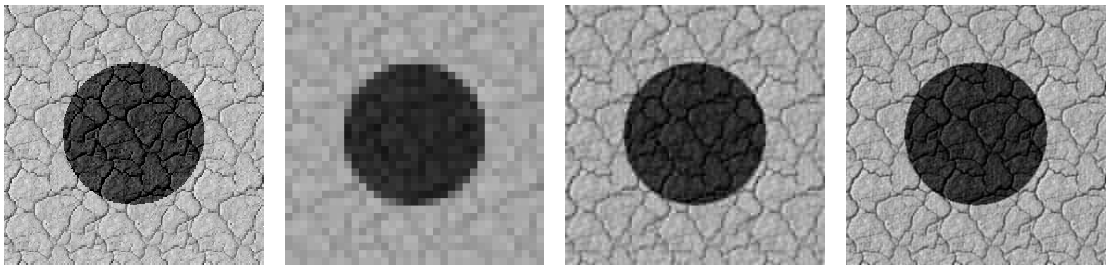
The resulting integers $\widetilde{e}_{nm}$ are stored following a zigzag path

$$\widetilde{e}_{00},\ \widetilde{e}_{10},\ \widetilde{e}_{01},\ \widetilde{e}_{11},\ \widetilde{e}_{20},\ \widetilde{e}_{30},\ \widetilde{e}_{12},\ldots,\ \widetilde{e}_{77}. \tag{2.76}$$



This finite sequence is very likely to be few numbers followed by a majority of zeros. It is easy to understand that a file with a lot of zeros admits an efficient lossless compression. We do not enter into the details of the compression here. We only mention that even a primitive method like RLE (Run Length Encoding) applied to the zeros i.e., substituting strings of zeros by their lengths, gives a noticeable result. In the case of JPEG, RLE is combined with other method (in practice *Huffman encoding*).

Perhaps you think that assuming that turning most of the a_{nm}'s into zeros without losing a lot in quality is wishful thinking. If you do some experiments with no quantization, just cropping the discrete Fourier expansion (2.72), it will amaze you how much information can be removed with little visual impact. If you do not want to do any experiment you can still look at the following 280×280 images:



original $\quad a_{nm}=0,\ n+m\geq 1 \quad a_{nm}=0,\ n+m\geq 3 \quad a_{nm}=0,\ n+m\geq 6$

In the second image we only keep a_{00} and then we see the blocks as thick pixels. In the third figure we keep 6 Fourier coefficients, it is less than 10% of the information, and we have a faithful idea about the original. Finally, in the last image with about 33% of the information we get something difficult to distinguish from the original. If you think that this example is made up, it is just the opposite because the furrows cause a bad performance of Fourier analysis. If you try a typical photo instead of this artificial small sized composition you will be more amazed.

Let us now consider the treatment of colors. In Computer Science, colors are commonly expressed by their coordinates in the *RGB color space*. These coordinates take values in $\{0, 1, 2, \ldots, 255\}$ and indicate how much of *R*ed, *G*reen and *B*lue contains the color. For instance, $(255, 0, 0)$ is pure red, $(0, 0, 0)$ is black, $(255, 255, 255)$ is white and in general (n, n, n) are all possible gray tones. In total there are $256^3 = 16777216$ colors. When the coordinates are nearby the colors are indistinguishable (at least by me) at naked eye. The choice of red, green and blue as primary colors is related to human perception and probably to the available technologies. As an aside, the difficulties to make cheap blue LEDs (light-emitting diodes) stopped during decades this promising technology for full color LED displays that we see everyday[4].

A problem with the RGB color space is that it is not uniform with respect to our perception. When the *brightness*, the arithmetic mean of the coordinates, is smaller we have more difficulties to distinguish colors and it also depend on the color, for instance, we are less sensitive to changes in blue. In different applications one change the RGB color space by other color spaces (arguably the most known after RGB is CMYK used in professional printing). In JPEG and in many applications related to video, the color space is $\mathrm{YC_bC_r}$, where Y is the *luminance* and the pair $(\mathrm{C_b}, \mathrm{C_r})$ is the *chrominance* (or both are the *chroma components*). The relation between both color spaces is given by [Sal02, p.145]

$$\begin{pmatrix} \mathrm{Y} \\ \mathrm{C_b} \\ \mathrm{C_r} \end{pmatrix} = \begin{pmatrix} 0 \\ 128 \\ 128 \end{pmatrix} + \frac{1}{256} \begin{pmatrix} 77 & 150 & 29 \\ -44 & -87 & 131 \\ 131 & -110 & -21 \end{pmatrix} \begin{pmatrix} \mathrm{R} \\ \mathrm{G} \\ \mathrm{B} \end{pmatrix}. \tag{2.77}$$

In practice this relation is not so exact because we ajust the result to fit in a byte.

Roughly Y represent the achromatic brightness of an image for our perception and $\mathrm{C_b}$ and $\mathrm{C_r}$ the color information. It turns out that we are much more sensitive to Y than to chrominance (in fact for chrominance usually only a part of the pixels are considered, see the comments below).

For B/W images the RGB colors are (n, n, n) and (2.77) implies $\mathrm{Y} = n$, $\mathrm{C_b} = 128$ and $\mathrm{C_r} = 128$, showing that Y carries the achromatic information. After a normalization subtracting 128, the chroma components become zero. The Y component is processed as described above. For the other components, one repeats the procedure but in a careless way, meaning that we take instead of (2.75) a quantization matrix with in general larger values. The recommendation in [CCI91, Annex K] is

$$Q = \begin{pmatrix} 17 & 18 & 24 & 47 & 99 & 99 & 99 & 99 \\ 18 & 21 & 26 & 66 & 99 & 99 & 99 & 99 \\ 24 & 26 & 56 & 99 & 99 & 99 & 99 & 99 \\ 47 & 66 & 99 & 99 & 99 & 99 & 99 & 99 \\ 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \\ 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \\ 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \\ 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \end{pmatrix}. \tag{2.78}$$

The outcome is that the corresponding values $\widetilde{e}_{nm}$ to be stored are more likely to be zero than the corresponding one for Y.

[4]The scientists that created the first blue LEDs were awarded with the Nobel prize in 2014.

The lines below show the main aspects of the algorithm but the implementation involves many other technical points that probably will be of interest to you only if you have an itch for engineering. We mention here three of them. Firstly, not too much precision is need when computing a_{nm} because we are going to quantize the results, then one can speed up the $64W \times H$ operations using different formats of numbers. When exporting an image as JPEG with GIMP, you have some control on it in `Advanced options>DCT method`. Secondly, a_{00} is typically very large in comparison with the rest of the values because it gives the sum of the colors of the block. To avoid to store large numbers after quantization, it is normalized subtracting certain quantity. This coefficient is called the *DC component* and the rest the *AC component*, following the electrical terminology AC/DC (Alternating Current/Direct Current). A third technical point is that as C_b and C_r are less important than Y, when treating the former components it is usual to take into account only one out of each two or each four pixels. In GIMP you can choose among four possibilities in `Advanced options>subsampling`.

If we examine critically the format, the subdivision into 8×8 blocks to apply Fourier analysis on them is somewhat arbitrary (and in part related to the capabilities of 1990s computers). It is a kind of balance, because taking very small blocks, 1×1 in the limit, is like doing nothing and choosing larger blocks we take the risk of including sharp changes of the color that avoid the well behavior of Fourier analysis. Summing up, we would like to enlarge these blocks in zones with gradients and perhaps to reduce them in zones with edges.

From the theoretical point of view, it would be convenient to have a so to speak adaptable Fourier analysis able to work at different scales. This theoretical aim is achieved with wavelets as we shall see in the next chapter. The idea led to create in 2000 the JPEG2000 format that employs wavelet transforms. Although it is superior to JPEG it has not been successful and very seldom cameras, scanners, browsers and widespread software support it. One can cook many explanations for it, a sound one is that it appeared when JPEG was widely used (JPEG2000 is not backward compatible) and with the features of modern computers more compression or more quality is no longer a primary issue (see the final comments in [Aus08]).

Suggested Readings. You can find information about JPEG in the short note [Aus08] and in the books [Sal02] and [GWE03, §8.5.1]. The website of the Joint Photographic Experts Group `https://jpeg.org/` contains references and information. If you like to go to the source without reading long documentation files, you will enjoy the 1991 paper [Wal91].