An exercise on JPEG compression evaluation

Roberta Prendin
819575

Multimedia Systems course, A.A. 2014-2015

Contents

1 The JPEG standard

JPEG is a **modern image compression standard** developed by the *Joint Photography Expert Group* since 1986 and accepted as an **international standard in 1992** [LD 2004]. Widely considered as the most commonly used format for storing photographic images, the JPEG standard (ISO/IEC 10918) is composed by **six separate specifications** [JO 2015]:

1. JPEG Requirements and guidelines (ISO/IEC IS 10918-1, ITU-T T.81)
2. JPEG Compliance testing (ISO/IEC IS 10918-2, ITU-T T.83)
3. JPEG Extensions (ISO/IEC IS 10918-3, ITU-T T.84)
4. JPEG Registration authorities (ISO/IEC IS 10918-4, ITU-T T.86)
5. JPEG File Interchange Format (JFIF) (ISO/IEC IS 10918-5, ITU-T T.871)
6. JPEG Application to printing systems (ISO/IEC IS 10918-6)

Interestingly, the JPEG standard refers to a **class of compression algorithms** and not to a specific file format. In fact, in order to produce files with embedded JPEG streams a number of file format standards have been developed, the most common being JPEG/Exif (*Exchangeable image file format)*, used by photographic image capture devices and JPEG/JFIF, used instead for storing and transmitting photographic images on the web [Exiv2 2015]. The most important feature of the JPEG standard is that it reduces the size of the output image by taking advantage of the **natural limitations of the human vision**: since our perception of brightness is usually more accurate than that of colours, **JPEG represents chrominance information with less precision** suppressing unnoticeable details and thus reducing the file size without compromising the overall quality of the image.

1.1 JPEG compression

The original JPEG standard defines **four compression modes** — hierarchical, progressive, sequential, and lossless — each with multiple encoding processes.

- **Sequential compression mode**: the plain JPEG compression mode. As the name suggests, images are simply scanned from the top to the bottom: each colour component is completely encoded in a single scan, with 8 or 12 bits of precision. As for the encoding processes, we can either use Huffman encoding or arithmetic encoding. A subset of the sequential mode with Huffman encoding is the so-called *baseline* process, where images are scanned with 8 bits of precision and are restricted to fewer Huffman and quantization tables [JM 1999].

- **Progressive compression mode**: each colour component of the image is encoded in multiple scans, from a minimum of 2 to a maximum of 896 per component. The first few scans produce a rough version of the image while other scans refine it: this allows the user to view images while they are being decoded — a particularly useful feature when pictures are being downloaded over a slow network connection.

- **Hierarchical compression mode**: a variant of the progressive compression mode, images are broken down into a number of *frames* — collections of one or more scans where the first frame creates a low-resolution version of the image and the remaining frames refine it by increasing the resolution. The drawback of this compression mode lies in its complexity since it requires much more processing than the other modes: as such, it is not generally used.

- **Lossless compression mode**: the original JPEG standard also defined a lossless compression mode that aimed to preserve the original image. However, for most applications it did not work as well as other available formats and thus it was never widely used. A new lossless compression mode called `JPEG-LS` was created afterwards.

As we can see, the JPEG standard is extremely vast. In this assignment we will focus on *sequential baseline* and *progressive* JPEG files with Huffman coding and 8-bit samples.

1.2 JPEG Compression Algorithm

Suppose we have a high-resolution image we want to save as a JPEG file. The JPEG compression algorithm is composed by the following steps:

- **Colour space conversion**. The original image is preprocessed, undergoing a colour space conversion from RGB (*Red*, *Green*, *Blue* components) to YCbCr (*Luminance*, *Chrominance blue*, *Chrominance red* components). This way, the luminance channel is separated from chrominance channels. In this step *chroma subsampling* can also be performed depending on the compression quality chosen by the user.

- **Decomposition into blocks**. The image is decomposed into 8×8 blocks of pixels called *Minimum Coded Unit* (MCU); if an image doesn't have enough pixels in a row or column to complete a MCU either the image is resized or a *partial MCU* is used. Note that when a JPEG file shows block artefacts they are actually **visual discontinuities between one or more of these blocks** caused by the high level of compression chosen by the user.

- **Discrete Cosine Transformation (DCT)**. The image is converted from a *spatial* domain representation to a *frequency* domain representation: DCT, in fact, expresses the contents of each MCU block as a sum of cosine functions oscillating at different frequencies, and as such it is applied left to right, top to bottom to each block. The result is that each 8×8 block contains 64 DCT coefficients, which can be manipulated for compression in the next steps of the algorithm [SDC 2015]. An example of DCT on a MCU is shown below (fig. 1). Note that the upper left coefficient of the block aggregates most of the signal, defining the basic hue of the whole block, and is called DC (*direct current*) coefficient; the remaining 63 coefficients are called instead *AC coefficients*, where AC stands for *alternating components*; they become lower in magnitude as they move farther from the DC coefficients [SDC 2015].

- **Quantization**. As we have said before, the human eye can tell small differences in brightness over a relatively large portion of an image (low-frequency details); the same does not apply when smaller areas are concerned: high-frequency details are not easily spotted. The JPEG algorithm takes advantage of these limitations discarding many of these high-frequency details and

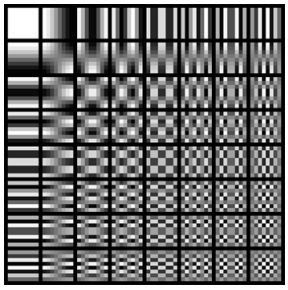

Figure 1: DCT on a 8×8 block [WMC-DCT 2015]

preserving instead the slowly-changing image information [IA 2015]. This is achieved by dividing each DCT coefficient by a corresponding value using a quantization table and then rounding each result to the nearest integer. In particular, a large quantizing factor is chosen for high frequencies, which will likely be rounded to zero. As a general rule, the higher the quality setting chosen by the user the smaller the coefficients in the quantization table. Note that quantization tables are not standardized, meaning they vary between digital cameras, softwares and applications.

- **Zig-zag organization**. In this step the previous matrix is reordered from the top-left corner to the bottom-right corner, producing a **64-element vector** whose elements are basically sorted from low-frequency components to high-frequency components (fig. 2). Since high-frequency components are likely reduced to zero, this vector will typically have a sequence of many zeros at the end depending on the quantization table used.

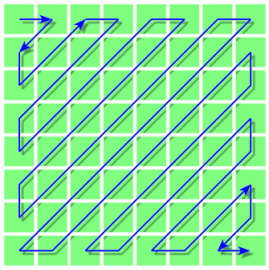

Figure 2: Zig-zag scan on a 8×8 block [WMC-ZZ 2015]

- **DPCM on DC component**. As a general rule, the DC component value

is always different for each block, but its value is often close to that of the previous block [RY 2015]. The *Differential Pulse Code Modulation* encodes exactly this difference between DC components.

- **RLE on AC components**. *Run Length Encoding* is a data compression technique used to encode sequences of identical values in an efficient way. Since the 64-element vector contains many zeros, RLE can be used to save the non-zero values and then count the number of zeros between these non-zero values; more in details, RLE stores a (*skip*, *value*) entry where *skip* is the number of zeros before the next non-zero component, *value*.

- **Huffman Coding / Entropy Coding**. Strings of values are represented using a much shorter code word, which is then listed into a dictionary called Huffman table. The more a string or a pattern is common, the shorter the corresponding code (only a few bits) whereas less frequent strings have longer codes. Since the dictionary is stored in the JPEG file, it is easy to lookup the encoded bit string to recover the original values.

1.3 JPEG Chroma Subsampling

Since the human eye is less sensitive to changes in chrominance than in luminance, the JPEG algorithm can also perform the so-called *chroma subsampling*, discarding further colour information. Chroma subsampling, in fact, is the process of **sampling the colour information of the image at a lower resolution** than the original [IA-CS 2015]: the more colour information is subsampled, the more artefacts are usually visible. With YCbCr we have different levels of chroma subsampling:

- **4:4:4 or** 1×1. Chroma components are sampled at the same rate as the luma component, meaning that there's no chroma subsampling.

- **4:2:2 or** 2×1. Chroma components are sampled at half the rate of the luma component in the vertical direction. This solution is common for digital cameras.

- **4:1:1**. Chroma components are sampled at a quarter of the rate of the luma component in the vertical direction.

- **4:2:0 or** 2×2. Chroma components are sampled at half the rate as the luma component in both the horizontal and vertical directions.

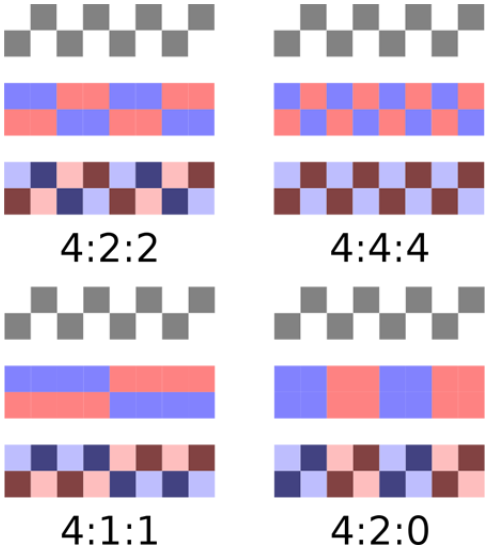

Figure 3: examples of chroma subsampling [WMC-CS 2015]

1.4 Structure of a JPEG file

Any JPEG file can be broken into a series of components delimited by *markers* — 2 bytes where the first one is always `0xFF` while the other is instead a code that identifies the type of the marker. As a general rule, this second byte must never be `0x00` or `0xFF`. Depending on their payload, markers can be organized into two general groups:

- **Stand-alone markers**. They consist of no payload other than their 2 bytes.
- **Markers with payload**. They are followed immediately by a 2-byte value that states the length of the segment's payload, these two bytes included.

  ```
  0xFF 0xXX size1 size2 data bytes
  ```

  The length of the payload can be computed as `length` = `size1` $*$ `256` + `size2`, which means that the actual content of the segment is `length - 2` bytes long.

The most important markers are the following:

Marker	Bytes	Payload	Meaning
SOI	0xFF, 0xD8	-	Start Of Image
RSTn	0xFF, 0xDn	-	Restart
EOI	0xFF, 0xD9	-	End Of File
SOF0	0xFF, 0xC0	variable size	Start Of Frame (Baseline)
SOF1	0xFF, 0xC1	variable size	Start Of Frame (Sequential)
SOF2	0xFF, 0xC2	variable size	Start Of Frame (Progressive)
SOF3	0xFF, 0xC3	variable size	Start Of Frame (Lossless)
DHT	0xFF, 0xC4	variable size	Define Huffman Table(s)
DQT	0xFF, 0xDB	variable size	Define Quantization Table(s)
DRI	0xFF, 0xDD	4 bytes	Define Restart Interval
SOS	0xFF, 0xDA	variable size	Start Of Scan
APPn	0xFF, 0xEn	variable size	Application-specific
COM	0xFF, 0xFE	variable size	Comment

Table 1: Markers of any JPEG stream

As we can see, JPEG files can use many different markers; in this tutorial we will focus on a subset of them to analyse our samples:

- DHT. The DHT marker defines one or more Huffman tables, which are identified by a class (AC or DC) and a number. Baseline mode is limited to two of each type while progressive and sequential modes are limited instead to four [JM 1999].
- DQT. The DQT marker defines one or more quantization tables used in the file, up to 4 different tables.

2 Practical examples of JPEG compression

We will now consider two of the most famous raster graphic editors —Adobe Photoshop and GIMP — and see how they compress a series of very different

images. The original high-resolution files we will use in this tutorial are[1]:

- **Geometric pictures** composed of polygons, with no shadows and a variable amount of colours. The analysis will focus on how the very sharp edges of these samples — GNS2C (Geometric No Shadows 2 Colours) and GNS*C (Geometric No Shadows * Colours) from now on — are compressed.

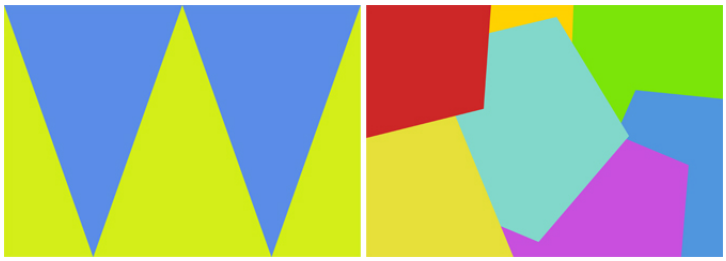

Figure 4: GNS2C (left) and GNS*C

- **High-resolution photos**: two landscape pictures shot with two different cameras — a Nikon D300s for the first sample, called LEP (Landscape Extended Palette), and a Nikon D80 for the second, called LLP (Landscape Limited Palette). Both samples have rich, complex shadows (LEP was the result of a long exposition which explains the noise the sample shows) but very different colour palettes. The analysis will focus on how areas with smooth surfaces (LEP lake), vibrant colours (LEP sky) and soft tones (LLP clouds) are compressed.

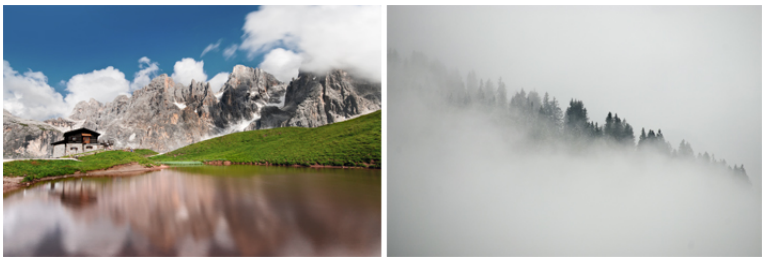

Figure 5: LEP (left) and LLP

- **Solid and gradient pictures** with two (S2, S2S) or three (S3, S3S) different colours either juxtaposed or gradually blended together. The analysis will

[1]These samples are available in `/Materials/Original samples`.

focus on how each application compresses the areas where colours come in contact.

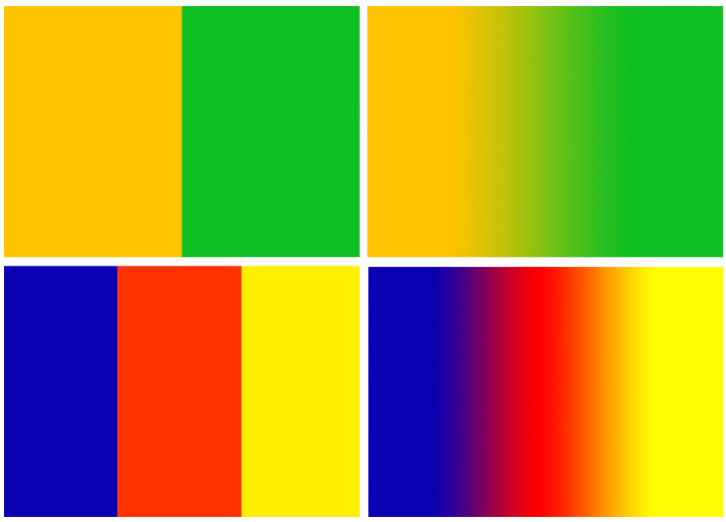

Figure 6: S2 and S2S (top), S3 and S3S (bottom)

- **Pictures with text**: pictures with black text (few words of the classic *lorem ipsum* written using *Adobe Garamond Pro*) on a white background or white text on a black background.

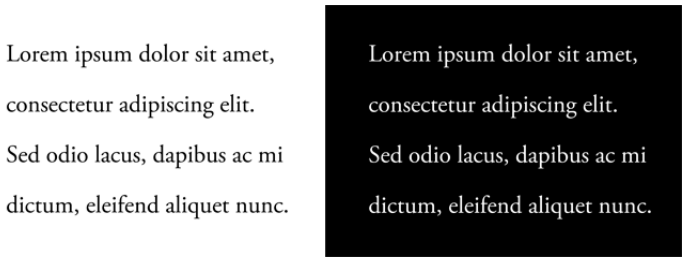

Figure 7: White (left) and Black textual images

As for the analysis itself, we will use the JPEG decoding utility `JPEGsnoop` to examine the characteristics of the JPEG compression and decode the various settings that were used by graphic editors. To visualize the differences between several compressions of the same picture we will use an image editor like Photoshop CS5 and perform the ***Error Level Analysis*** **(ELA)**. ELA is a technique generally

used to **highlight areas within an image that are at different compression rates**: areas with uniform colouring (a white wall, a blue sky, ...) usually have a lower ELA result (darker colour) than high-contrast edges (lighter colour). ELA is particularly useful in many scenarios (for example to see whether an images was digitally altered) but in this tutorial it will be used to visualize the differences between several compressions of the same sample. ELA can be implemented in Photoshop CS5 as follows:

1. Open the high-definition file and one of its corresponding JPEG versions;
2. Superimpose the JPEG version on the original file as a new "subtract" layer;
3. Manipulate a new "Levels" layer to see the differences between images.

3 Adobe Photoshop JPEG compression

Adobe Photoshop is a raster graphic editor developed in 1987 by Thomas and John Knoll, who later sold the distribution licence to Adobe Systems Incorporated. Since then Photoshop has become the *de facto* standard for raster graphic editing, supporting a vast number of file formats in addition to its own `Photoshop Document` (`PSD`) and `Photoshop Big` (`PSB`) ones. Photoshop is also capable of rendering and editing text, vector graphics, 3D graphics and video: its set of features can be enhanced either using Photoshop *plug-ins* — third party applications that can run inside Photoshop — or automated by using Photoshop *actions*. The latest available version is the *CC 2015*, released on 15 June 2015, but in this assignment we will focus on an earlier version, *CS5*, launched in 2010. CS5 offers **two different methods** to save images: `Save as` and `Save for web`.

Save as

`Save as` is the standard method used by most Photoshop users to generate their final image files. It supports a very large number of output file formats and, since it is not optimized for transmissions over a network, it encodes additional information such as EXIF metadata (camera information, shot details, ...) slightly increasing the resulting file size. While the

size of the additional data might look insignificant in most cases, when trying to compress small images for a slow network this extra overhead can be burdensome.

Save for web

First introduced in 2005 with Photoshop CS2, `Save for web` supports a more limited number of output image file formats (`JPEG`, `GIF`, `PNG-8`, `PNG-24` and `WBMP`). Geared towards the compression of images that will later be displayed on the web, the goal of `Save for web` is to optimize as much as possible the size of the file without reducing the overall quality of the image. To this end, `Save for web` implements an interactive *live preview* of the JPEG compression quality sliders and allows the removal all unnecessary metadata in the output file, including copyright, EXIF data, time/date of the photo and any optional marker.

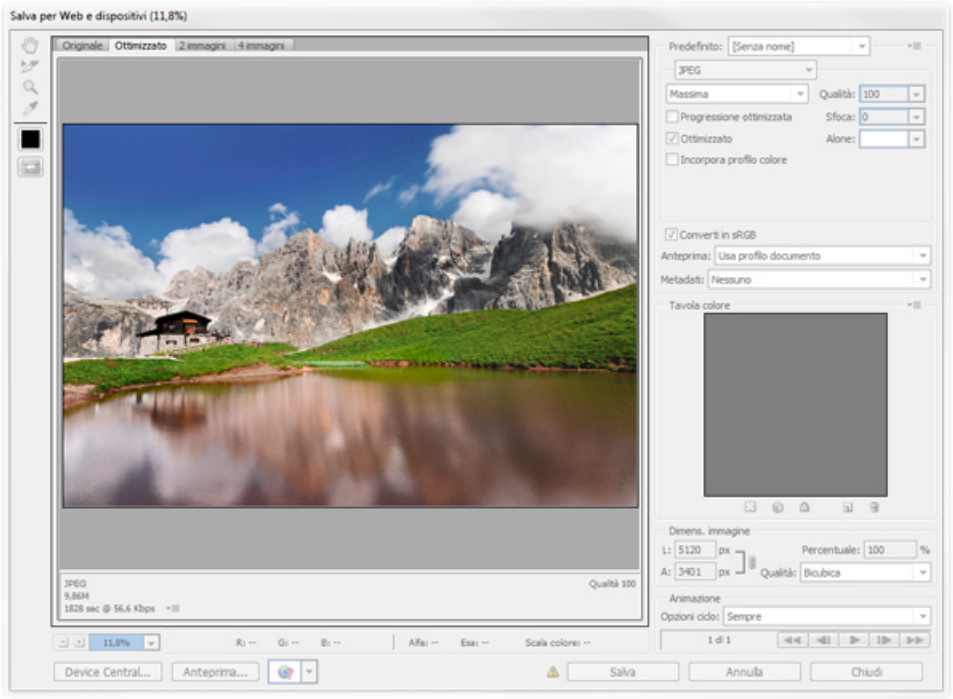

Figure 8: `Save for web` in Photoshop CS5 (Italian version)

Photoshop offers many different parameters to optimize JPEG compression — in fact, they are so numerous that most of their combinations will not be analysed in this tutorial. In the pages to come we will focus instead on the following settings:

- `Save as`: for each high-definition image we will analyse 13 standard baseline JPEG images. The compression levels we will analyse will be 0 (highest compression), 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11 and 12 (lowest compression). We will also see what happens when *baseline optimized* and *progressive optimized* compressions are chosen instead.

- `Save for web`: for each sample we will analyse 12 standard baseline non-optimized JPEG images, with quality set at 0 (highest compression), 10, 20, 30, 40, 50, 51, 60, 70, 80, 90, 100 (lowest compression). We will also see what happens when the *optimized* option is selected.

3.1 Geometric images

3.1.1 Baseline `Save as` JPEG analysis

To proceed with the analysis we save 13 different samples of GNS2C and GNS*C with compression levels varying from 0 (highest possible compression) to 12 (lowest possible compression). To get a basic understanding of the resulting samples we use an image editor of our choice and zoom first at 2000% and then at 200% on any edge we like. We collect the following results:

- **GNS2C**. Compression level 12 yields the best results: edges are so sharp that they can be easily confused with those belonging to the original high-definition image. The only downside is of course the file size, which is a little more than 1MB. Compression levels 11-7 still produce acceptable samples with sharp or mostly sharp edges and a limited amount of discolourations; unfortunately the same does not apply to the remaining compression levels where block artefacts, stained areas and discolourations on the edges are more and more prominent. To sum up, the more the image is compressed the more the overall quality of the samples deteriorates: zooming on each image reveals that edges appear progressively more blurry and ruined (fig. 6).

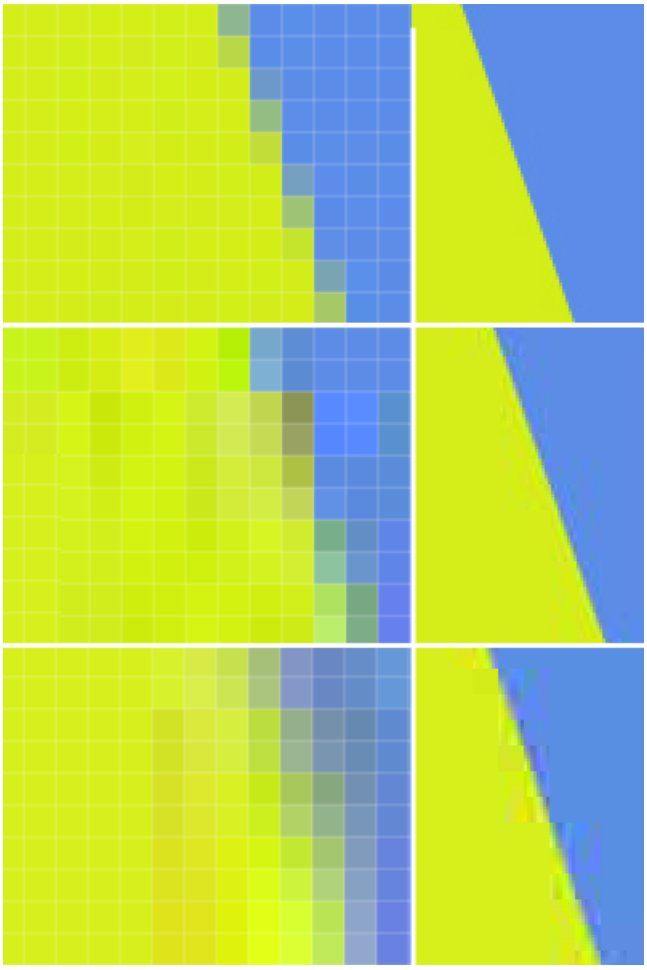

Figure 9: GNS2C compressed at 12 (top), 6, 0 (bottom), zooming at 2000% (left) and 200% (right)

- **GNS*C**. The previous considerations apply to GNS*C, where edges look less and less well-defined the more the original sample is compressed. 8×8 block artefacts are not visible when high compression levels like 12, 11 or 10 are selected but become more and more prominent the higher we compress, ruining the overall quality of the picture when the lowest compression levels (3-0) are selected.

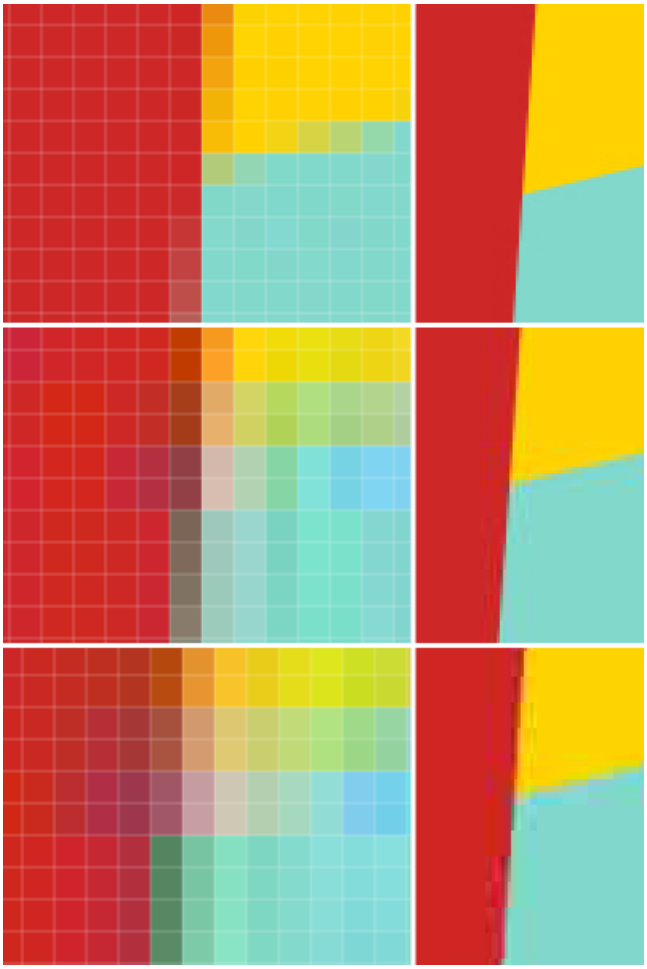

Figure 10: GNS*C compressed at 12 (top), 6, 0 (bottom), zooming at 2000% (left) and 200% (right)

Differences between each compression can be seen using the `ELA-GNS2C.tif` and `ELA-GNSC.tif` interactive files[2]. Unsurprisingly, we notice that the more we compress the original image the more differences we can spot; furthermore, block artefacts appear to be bigger in size and less well-defined the more the sample is compressed. As for the **file size**, JPEG compression yields samples whose sizes are very small indeed: given the original 37.1MB TIFF picture, we obtain a $\approx$ 1MB file at best (compression level 12) and a $\approx$ 170KB file at worst (compression level 0). There are also two interesting file size gaps between compressions 12-11

[2]Available in `Materials/Exploration in HD/Photoshop/Geometric & Shadows`.

and 7-6.

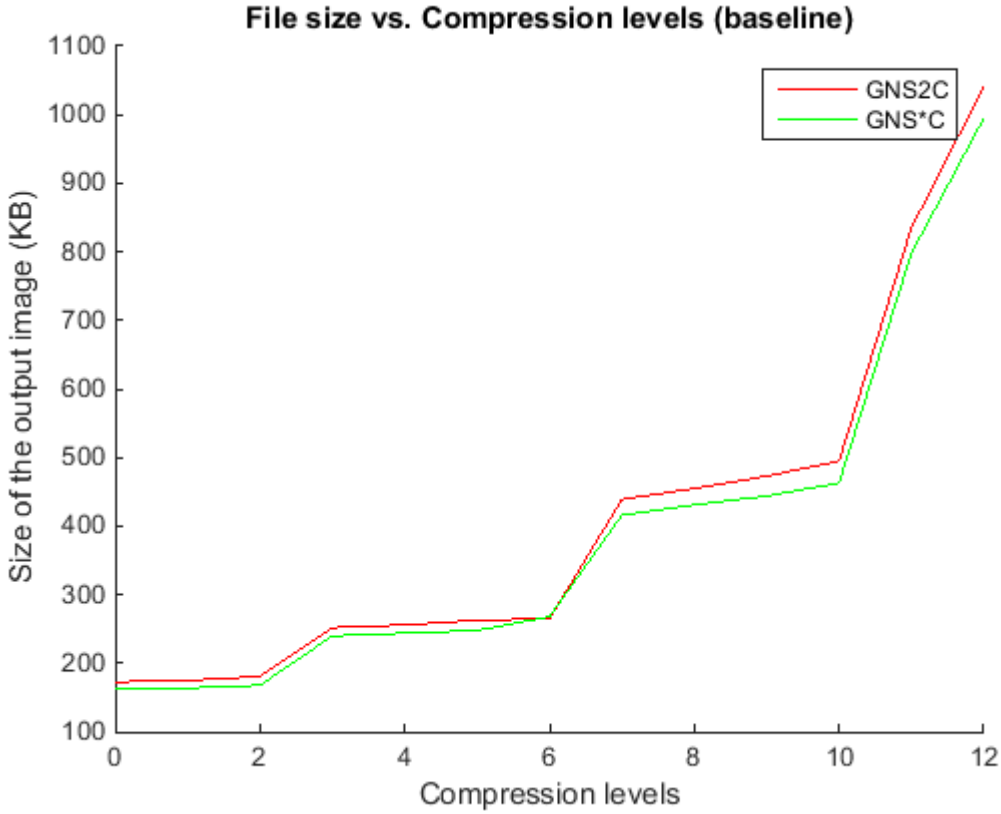

Figure 11: File size vs. compression levels. The data set is available as the stand-alone file called `GeometricDataSet - SaveAs.txt`

3.1.2 Baseline `Save for web` analysis

We will now focus on the other method offered by Photoshop, `Save for Web`. For simplicity's sake we will focus on compression qualities 100, 90, 80, 70, 60, 51, 50, 40, 30, 20, 10, 0, which are a good approximation of the whole set of 101 compression levels we can choose on the `Save for web` panel[3]. As before, to get a basic understanding of the compressions performed we can look at each picture

[3]The data set with 101 entries for GNS2C is available as a stand-alone file called `GNS2C - SaveForWebFullDataSet.txt`

using an image editor of our choice and zoom at 2000% and 200%. The **results are the comparable to the ones obtained using** `Save as`:

- **GNS2C**. Compression qualities 100 and 90 yield the best-looking results, showing very sharp edges. Again, the only downside appears to be the file size, which is roughly around 1MB for compression quality 100 and 823KB for compression quality 90.

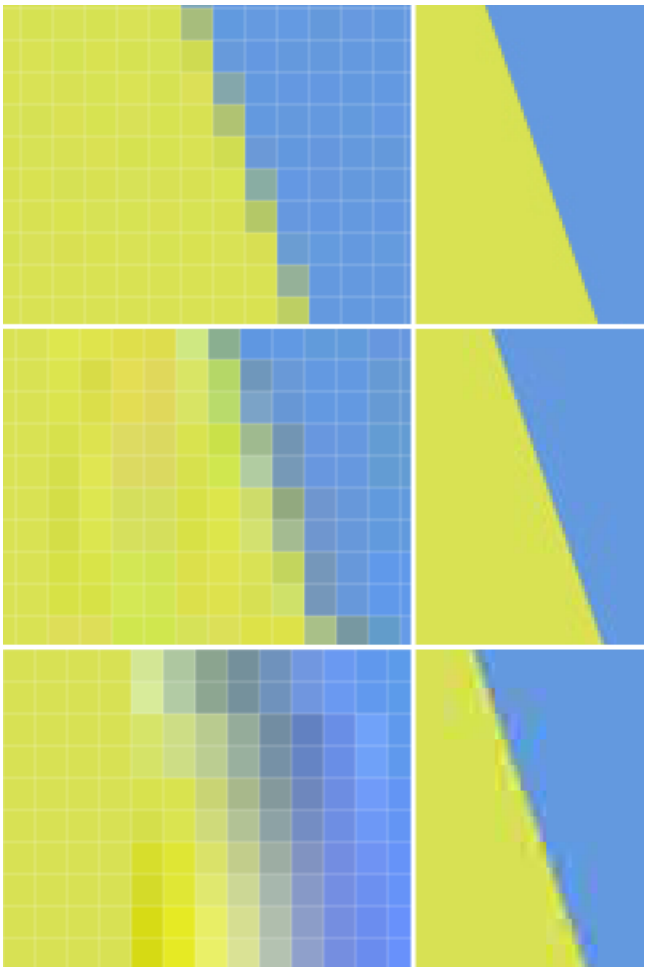

Figure 12: GNS2C compressed at 100 (top), 50, 0 (bottom), zooming at 2000% (left) and 200% (right)

The remaining compression qualities on the higher end of the range (80-51) produce acceptable results: block artefacts on the edges of the polygons

are visible but few and far in-between. If we select very poor compression qualities (50-0), however, it is easy to tell block artefacts and ruined areas on the edges of the polygons, even without zooming. Overall the lower the selected compression level the more the quality of the images deteriorates.

- **GNS*C**. As before, compression qualities 100 and 90 return samples with edges so sharp and well-defined that they can be easily compared to those belonging to the original high-definition pictures.

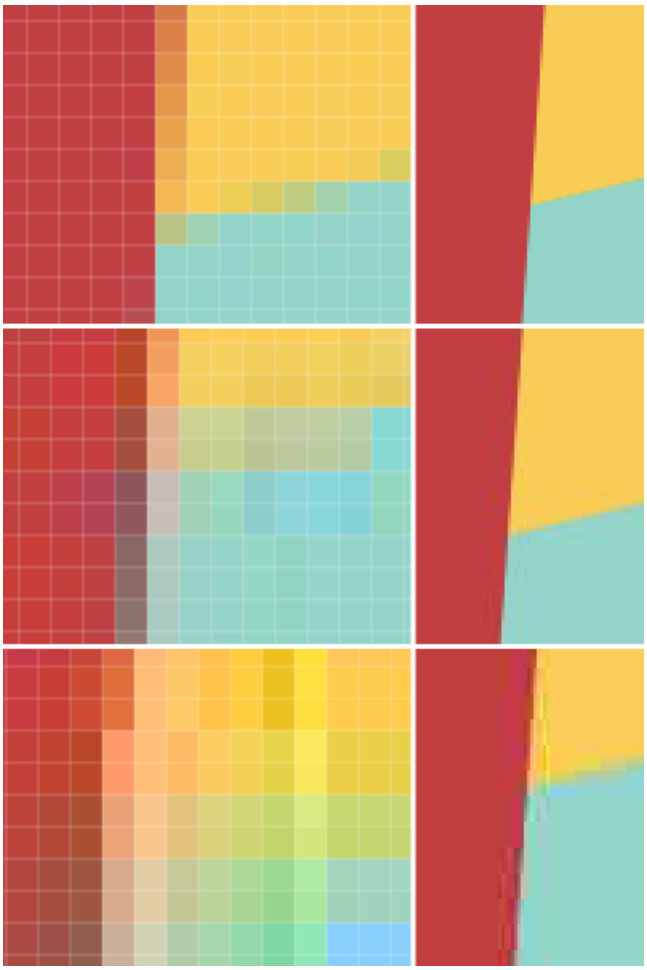

Figure 13: GNS*C compressed at 100 (top), 50, 0 (bottom), zooming at 2000% (left) and 200% (right)

Lowering the compression quality, however, we obtain pictures with very

visible block artefacts and discolourations on the edges of the polygons.

Visually, differences between each compression can be seen using the `ELA-GNS2C-SFW.tif` and `ELA-GNSC-SFW.tif` interactive files. The results are the same as before: the more we compress the original image the more blocks we can see on the edges of the polygons. As for the **file sizes**, as expected compression quality 100 returns the biggest files while compression quality 0 returns the smallest pictures; still, both compressions dramatically reduce the original file sizes, 37.1MB. Furthermore, there is an interesting file size gap between compression qualities 51 and 50.

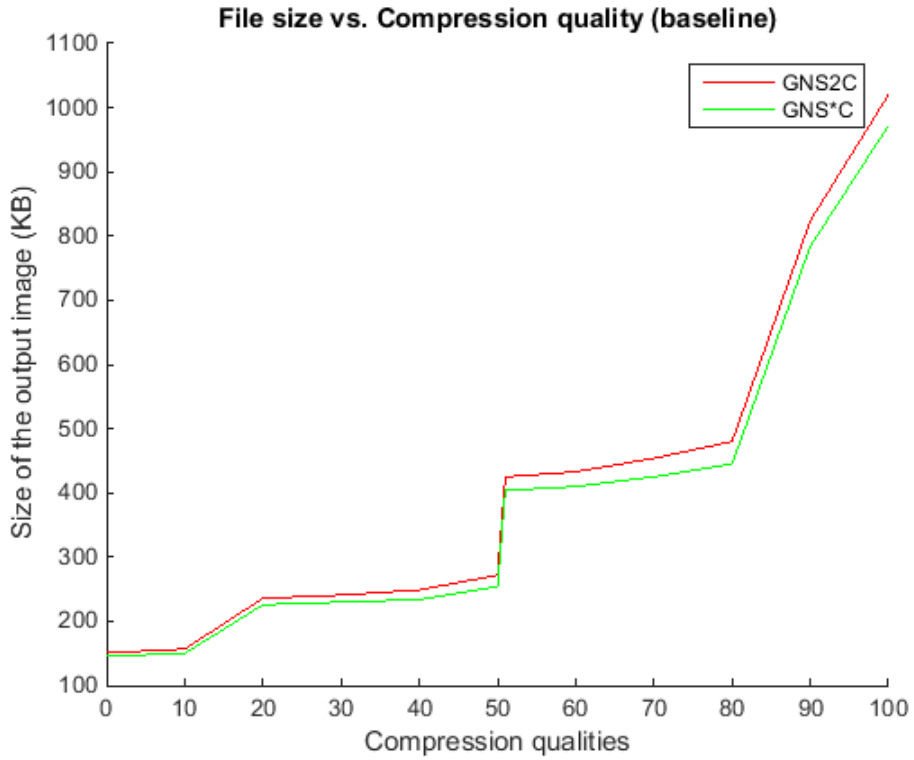

Figure 14: File size vs. compression qualities. The data set is available as a stand-alone file called `GeometricDataSet - SaveForWeb.txt`

3.1.3 Additional case study: geometric image with shadows

So far we have seen how Photoshop CS5 behaves when compressing images with simple polygons; what happens, however, if we add some shadows like in the example below?

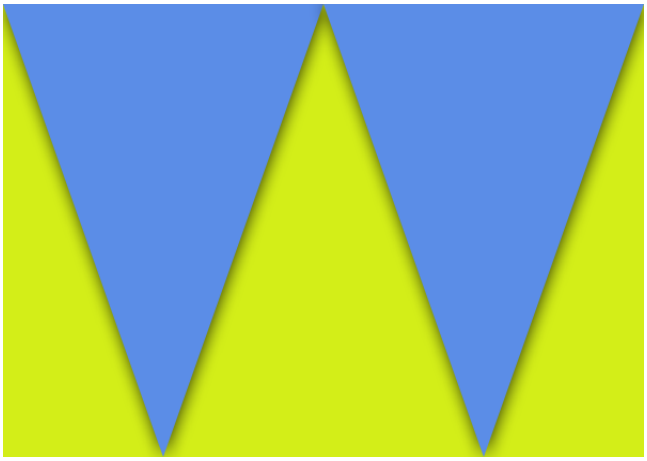

Figure 15: GS2C with sharp edges and shadows

We proceed as before using the `Save as` method, saving 13 different versions of GS2C with compression levels varying from 0 to 12. Then, using an image editor of our choice, we look at each sample: by zooming at 2000% and 200% we notice that the overall quality deteriorates significantly the more we compress each sample. **Shadows, as expected, are particularly critical**: while they look very smooth when the selected compression level is high enough (12, 11 or 10), they show a remarkable amount of block artefacts as soon as the compression level plummets to 6 or below. As for the **file size**, there appears to be a close relationship between GNS2C and GS2C samples with the same level of compression: when the compression level is high enough (at least 5) samples with shadows have a slightly bigger size, probably due to the complexity of rendering shadows smoothly; otherwise, samples with shadows have a slightly smaller size and show extremely visible block artefacts. More in detail, the comparison of GS2C and GNS2C file sizes can be summarized as follows:

- For compression levels 1-3 $GS2C < GNS2C$;
- For compression level 4 $GS2C = GNS2C$;

- For compression levels 5-12 $GS2C > GNS2C$.

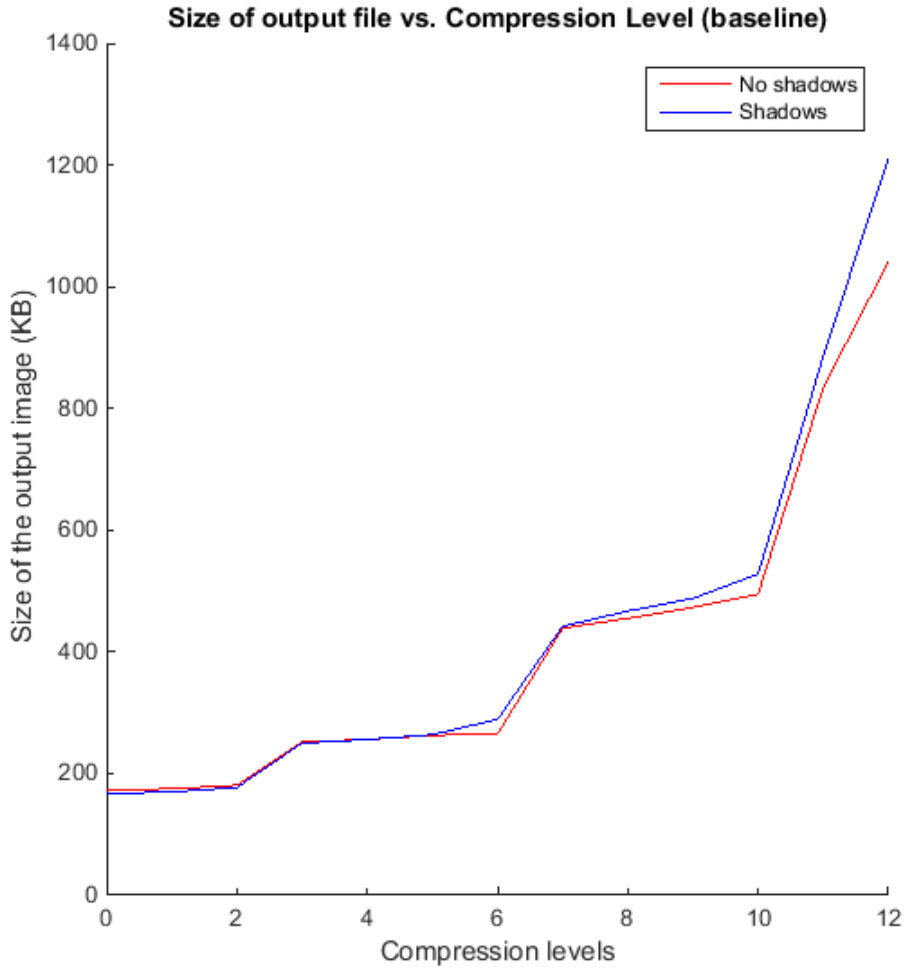

Figure 16: GS2C baseline "`Save as`" vs. GNS2C baseline "`Save as`" compressions. The data set is available as a stand-alone file called `GeometricDataSet - SaveAs.txt`

3.2 Pictorial images

3.2.1 Baseline `Save as` JPEG analysis

As before, we use the `Save as` function to obtain 13 different samples of both LEP and LLP, with compression levels varying from 0 to 12. Next, we evaluate each

compression sample using an image editor of our choice, zooming at 2000% and 200%. We obtain the following results:

- **LEP**. Unsurprisingly compression level 12 yields the best-looking results: colour transitions and shadows are smooth, edges are sharp, tones are not altered; the only downside is the bigger file size, which is around 10MB.

Figure 17: Compression level 12 vs. compression level 0 for LEP (top); artefacts of compression level 0 for LLP (bottom)

As for the other compressions, the lower the compression level the more block

artefacts and discolourations (areas where the transition between different colours is not smooth) we can notice on the samples. Compression level 6 appears to be the last acceptable compression level, with minor block artefacts and stained areas; at compression levels 5-0, instead, solid colour areas like the sky or the lake become more and more posterized, with block artefacts visible all over the picture (fig. 17).

- **LLP**. Compression levels 12 and 11 return the best-looking results: the trees are well-defined, shadows are smooth, the overall tone of the picture is not altered. In particular, compression level 11 represents a very good trade-off between file size (3MB) and overall quality of the image. Block artefacts begin to be noticeable at compression level 5 and become more and more visible the more we compress the original sample: at the highest compression levels clouds are severely posterized and discolourations are visible all over the picture (fig. 17).

Visually, differences in compression between each sample can be explored using the `ELA-LEP.tif` and `ELA-LLP.tif` interactive files; as expected, the more we compress the original high-definition file the more artefacts we introduce, with very visible posterized areas and ruined portions of the images at the highest compression levels (5-0).

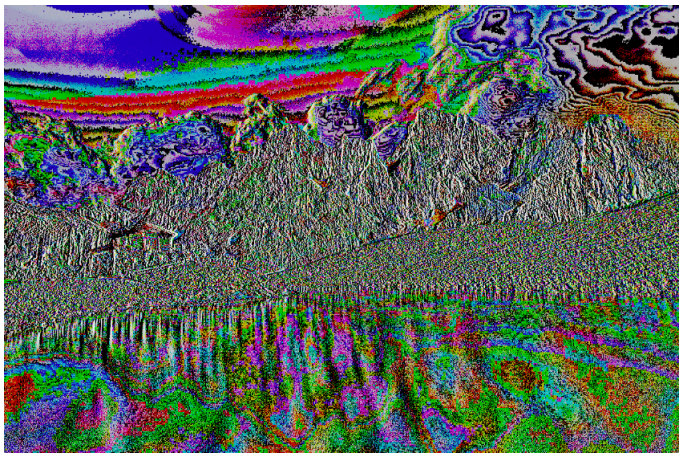

Figure 18: ELA of the compression level 0 for LEP

As for the **file size**, as usual compression level 12 yields the biggest files while compression level 0 returns the smallest images; the more we compress, in fact, the smaller the resulting sample. Note, however, that no matter the selected compression level the original size of the TIFF files — 125MB for LEP and 85.9MB for LLP — is always drastically reduced (fig. 19).

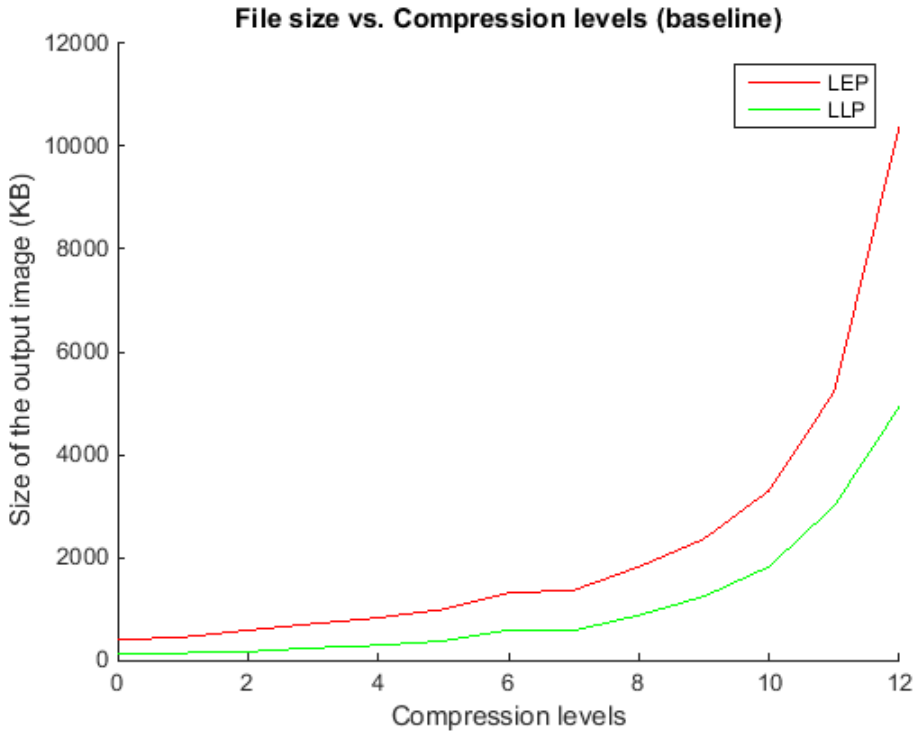

Figure 19: Compression levels vs. file size. The data set is available as a stand-alone file called `PictorialDataSet - SaveAs.txt`

3.2.2 Baseline `Save for web` JPEG analysis

Let's now see how `Save for web` fares when compressing pictorial images. As before, we save two sets of LEP and LLP images with compression qualities 100, 90, 80, 70, 60, 51, 50, 40, 30, 20, 10 and 0; next, we evaluate each sample using an image editor of our choice, zooming at 2000% and 200%. The results are, again, comparable to the `Save as` ones:

- **LEP**. Compression quality 100 returns the best-looking results, with smooth

shadows, sharp edges and realistic tones. As before, the only downside of this compression quality is the file size, which is around 10MB. Compression quality 90 still returns acceptable results, with slightly noticeable artefacts on the sky of the picture. As for the other compression qualities, the highest the compression the more block artefacts, posterized areas and stained portions of the images we notice.

- **LLP**. As before, compression quality 100 returns the best-looking results: the trees are well-defined and the overall tone of the image is correctly preserved. Still, the file size of this compression quality is rather big ($\approx$ 5MB). Interestingly, compression quality 90 seems to perform better for LLP than for LEP: the block artefacts we saw before are here unnoticeable. As for the other compression qualities, the more we compress the more we observe block artefacts and discolourations, especially at compression qualities 40-0.

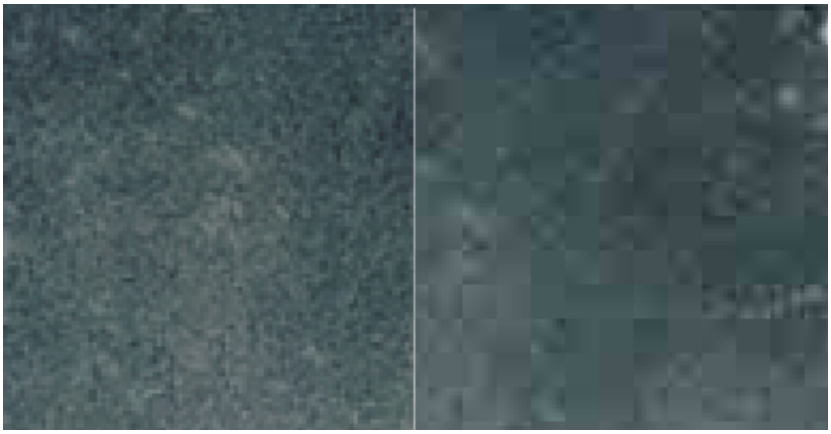

Figure 20: LLP compression qualities 100 (left) and 0 (right)

Visually, differences in compression between each sample can be explored using the `ELA-LEP-SFW.tif` and `ELA-LLP-SFW.tif` interactive files; as expected, the more we compress the original high-definition file the more artefacts (blunt transitions between colours, posterizations, discolourations, ...) we introduce.

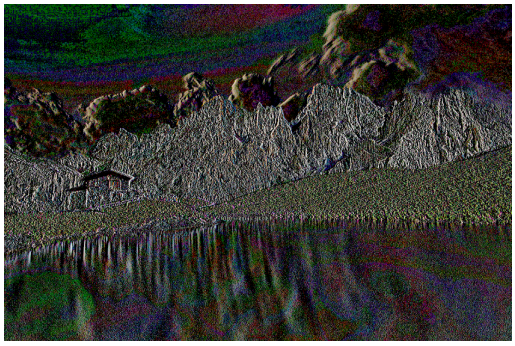

Figure 21: ELA of the compression quality 60 for LEP

As for the **file sizes**, the plot below (fig. 22) shows the expected behaviour: file size is directly proportional to the selected compression quality, with compression quality 100 returning the biggest files. Since the original TIFF files were 125MB (LEP) and 85.9MB (LLP) the JPEG compression always provides images with reduced sizes. Notice the small "bump" between compression qualities 50 and 51.

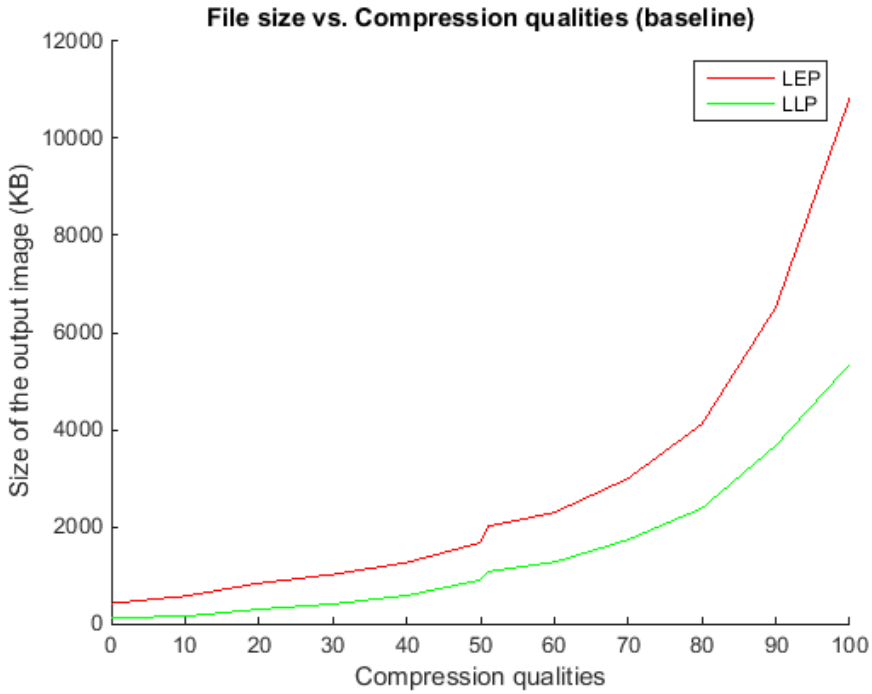

Figure 22: Compression qualities vs. file size. The data set is available as a stand-alone file called `PictorialDataSet - SaveForWeb.txt`

3.3 Solid and gradient images

3.3.1 Baseline `Save as` JPEG analysis

As before, we use the `Save as` function to obtain 13 different images of S2, S2S, S3 and S3S; then, by using an image editor, we evaluate each sample zooming at 2000% and 200%. The results are listed below:

- **S2, S3**. Compression levels 12 and 11 return the best-looking results and seem to be equivalent, with no visible colour bands or discolourations where different colours are juxtaposed. Every other compression, however, shows some degree of colour banding when zooming: different colours are not neatly separated.

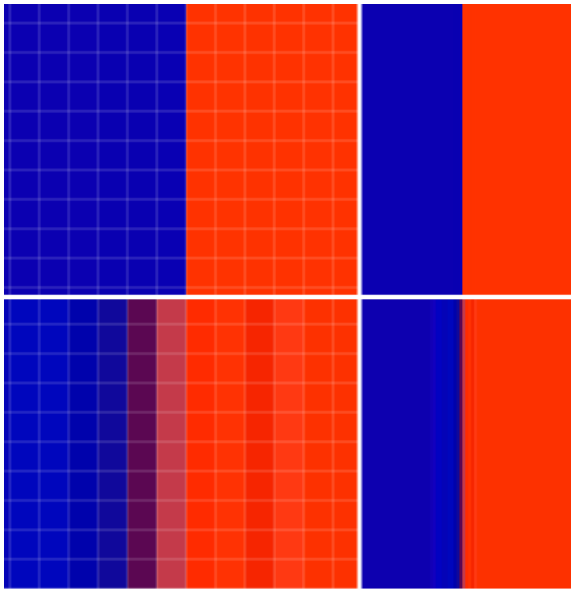

Figure 23: S3 compressed at level 12 (top) and 0 (bottom)

- **S2S, S3S**. Again, compression levels 12 and 11 yield the best-looking results, with no visible colour bands or artefacts on the blurred areas; on the contrary, every other compression level shows very visible colour bands. At the lowest compression levels the effect is very noticeable even without zooming, as the example below shows:

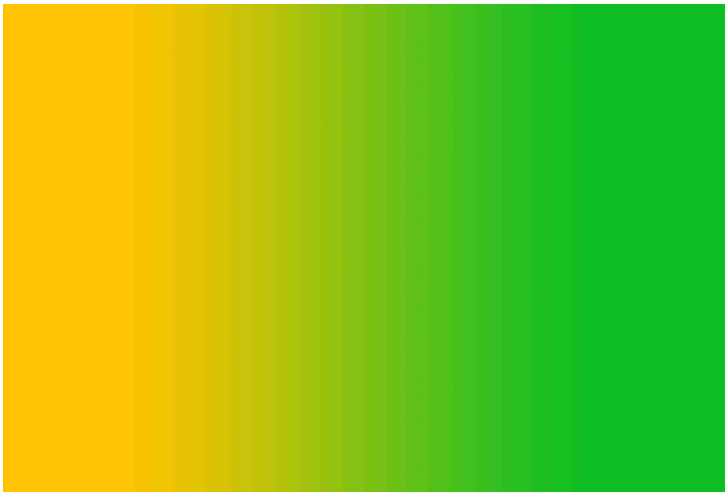

Figure 24: S2S compressed at level 0 with visible colour bands

Visually, these colour bands are especially visible if we consider the `ELA-S2.tif` and `ELA-S2S.tif` interactive files: the more we compress, the more colour bands and discolourations appear on the transitional areas of the pictures.

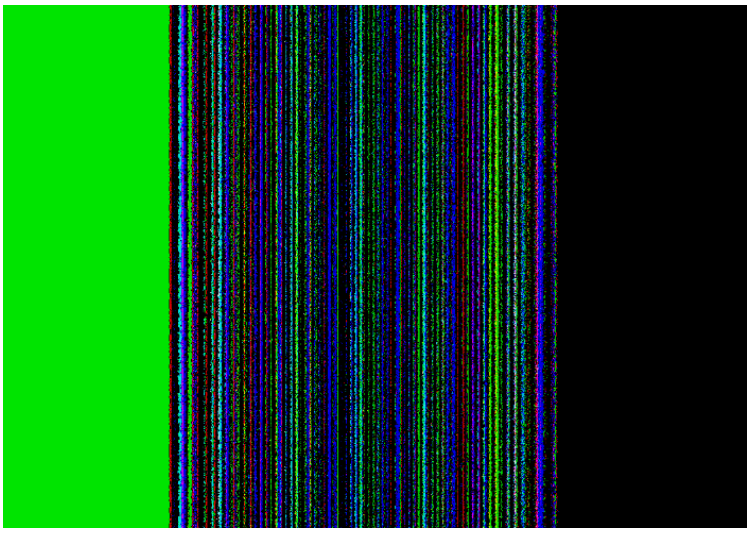

Figure 25: ELA for S2S

As for the **file size**, compression level 12 gives the biggest files exactly as noted before. File size rapidly decreases when higher compression rates are selected.

Notice as before the interesting file size gaps between compression levels 12-11 and 7-6.

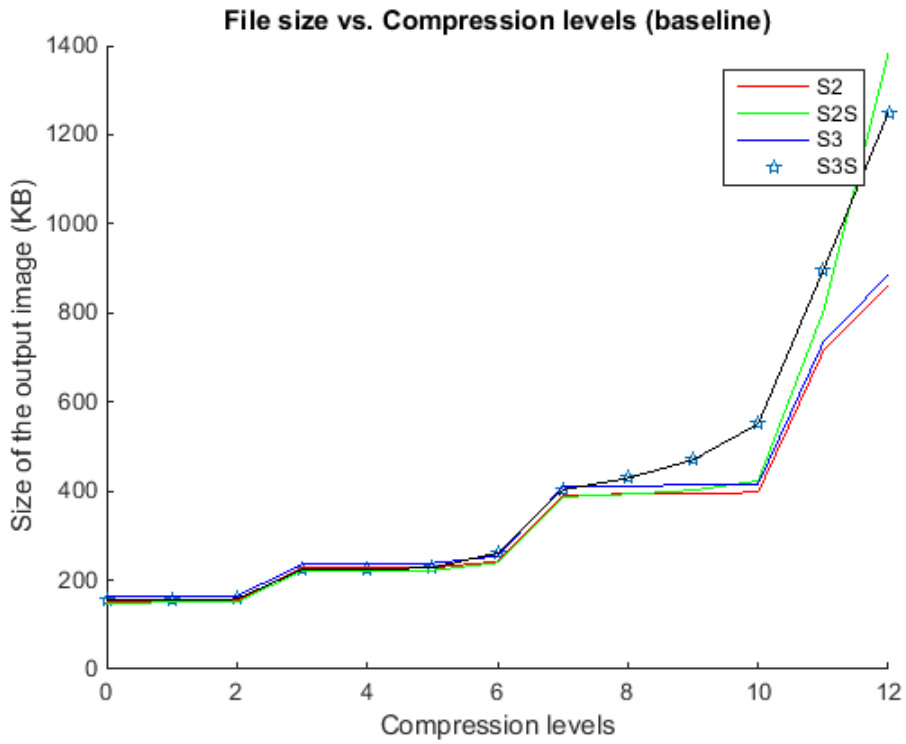

Figure 26: Compression levels vs. file size. The data set is available as a stand-alone file called `ColorBlockDataSet - SaveAs.txt`

Still, the sizes of the original high-definition pictures are strongly reduced: for example, if we consider compression level 12, we obtain the data shown below.

File	Original size	Compressed size
S2	40072KB	863KB
S2S	37434KB	1382KB
S3	88547KB	886KB
S3S	37433KB	1250KB

Note that compressed pictures with blended colours are considerably bigger in size when the selected compression level is at least 10 — a result that we noticed

even when we analysed the geometric picture with shadows. As we will see in the following pages, this is caused by the complexity of the image itself rather than by different compression approaches chosen by Photoshop CS5.

3.3.2 Baseline `Save for web` JPEG analysis

To analyse the `Save for web` function, we save four sets of S3, S3S, S2 and S2S images with level qualities 100, 90, 80, 70, 60, 51, 50, 40, 30, 20, 10 and 0; then, we visually evaluate these samples using an image editor. By zooming at 2000%, 200% and 100% we get the following results:

- **S2, S3**. Compression levels 100 and 90 seem to be equivalent, returning the best-looking results: colours are neatly juxtaposed and there are no block artefacts. In both cases some colour bands are present but they are visible only when zooming and completely unnoticeable otherwise. As for the remaining compression levels, we have more and more colour bands the more we compress the original high-definition image, with very visible discolourations when the selected compression level is between 30 and 0. Notice that the darker colour is usually the one showing very visible discolourations.

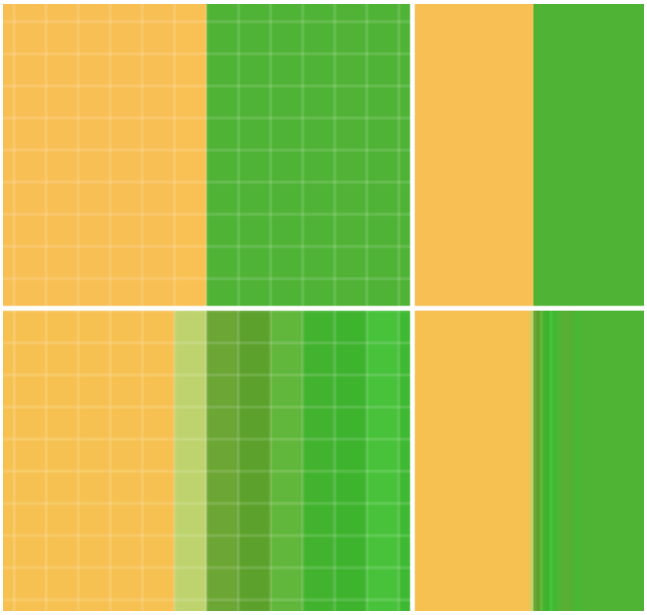

Figure 27: S2S compressed at levels 100 (top) and 0 (bottom)

- **S2S, S3S**. Again, compression levels 100 and 90 give the best-looking results, with uniformly blended blocks of colours. As expected, discolourations and colour bands are more and more evident when the selected compression level is very low, between 30 and 0. As before, the effect is very visible even without zooming on the picture (fig. 28).

Figure 28: S2S compressed at level 0 with visible colour bands

Visually, these colour bands are especially noticeable if we consider the `ELA-S2-SFW.tif` and `ELA-S2S-SFW.tif` interactive files: the more we compress, the more colour bands appear on most of the samples.

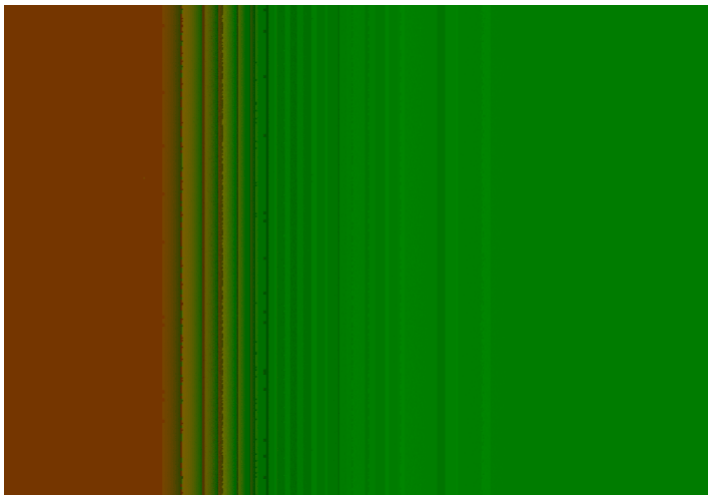

Figure 29: ELA for S2S at compression quality 0

As for the **file size**, the results are comparable to the ones obtained with the `Save as` features, as we can see from the following plot:

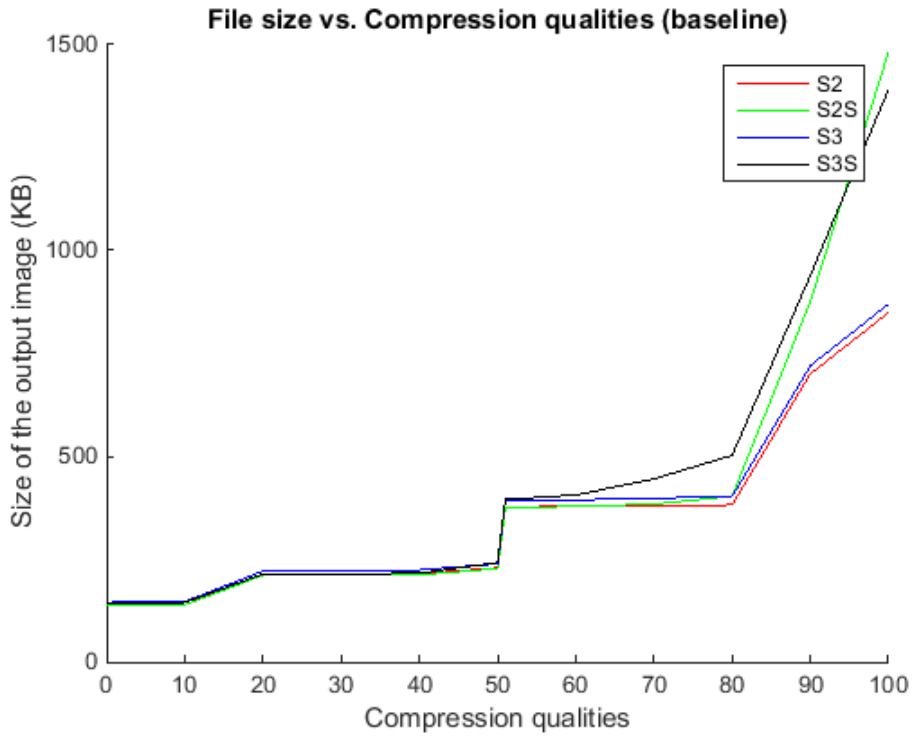

Figure 30: Compression qualities vs. file size. The data set is available as a stand-alone file called `ColorBlockDataSet - SaveForWeb.txt`

As usual, compression quality 100 returns the biggest files while the other compression qualities steadily reduce the sizes of the pictures. Notice that when the compression quality is low enough (55-0) the file sizes are more or less identical no matter the sample we consider. The same does not apply to the higher compression qualities (60-100), where the differences between samples are very noticeable. Furthermore, the interesting file gap between compression qualities 50 and 51 is still noticeable.

3.4 Textual images

3.4.1 Baseline `Save as` JPEG analysis

As before, we use the `Save as` function to obtain 13 different images of `Black` and `White`; then, we use an image editor of our choice to evaluate each sample, zooming at 200% and 2000% on each picture. The results are the following:

- **Black**. Compression levels between 12 and 9 yield the best-looking results, with no visible artefacts on the letters of the samples. Compression levels between 8 and 0 show instead more and more grey artefacts, reducing the overall quality of the image. Still, the text is always readable, no matter the selected compression level.

- **White**. Again, compression levels between 12 and 10 seem to give the best-looking results, with no artefacts on the background of the samples. The remaining compression levels, instead, show grey artefacts and very blurred areas. As before, the text is always readable no matter the selected compression level (fig. 31).

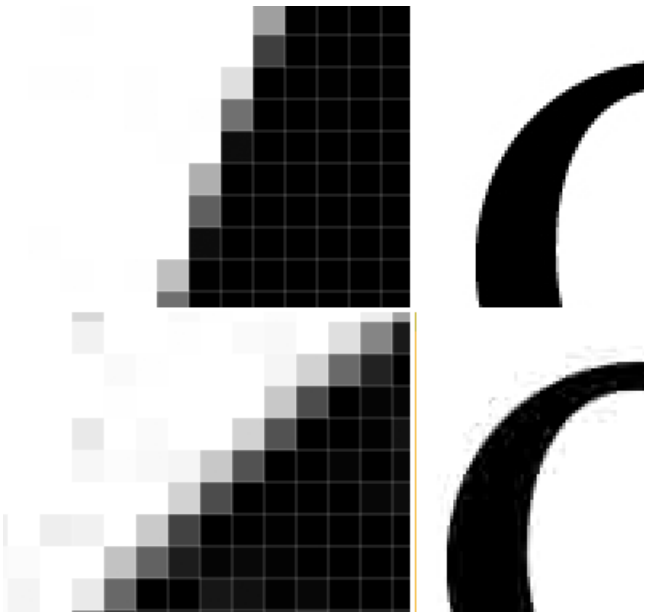

Figure 31: Compression level 12 (top) and 0 for `White`

Visually, colour artefacts are especially visible if we consider the `ELA-Black.tif` and `ELA-White.tif` interactive files: the more we compress, the more artefacts appear on the background (`White`) or on the letters (`Black`) of the samples. As for the **file size**, we obtain the following results:

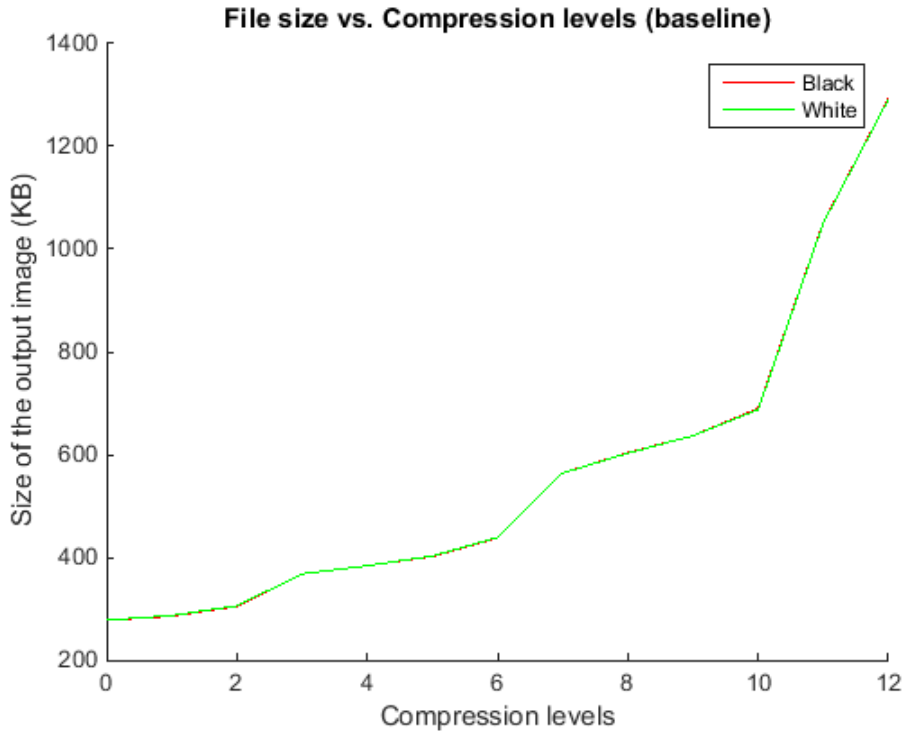

Figure 32: COmpression levels vs. file sizes. The data set is available as a stand-alone file called `TextDataSet - SaveAs.txt`

As expected the more we compress the original image the smaller the corresponding file size, with compression level 12 yielding the biggest files. Interestingly, the samples have **almost identical file sizes**, showing the usual file size gap between compression levels 12-11 and 7-6.

3.4.2 Baseline `Save for web` JPEG analysis

As for the `Save for web` function, we save two different sets of `Black` and White with level qualities 100, 90, 80, 70, 60, 51, 50, 40, 30, 20, 10 and 0; next, we visually evaluate each sample using an image editor of our choice, zooming at 2000% and

200%. The results are the following:

- **Black**. Compression qualities between 100 and 80 yield the best-looking samples, with no visible artefacts on the letters of the text. Every other compression quality shows instead more and more grey artefacts, reducing the overall quality of the image. Still, the text is always readable no matter the selected compression quality.
- **White**. Again, compression qualities between 100 and 80 seem to return the best results, with no artefacts on the background of the text. Other compression levels are not as remarkable, showing grey pixels and blurred areas. As before, the text is always readable no matter the selected compression level.

Visually, artefacts are very clear if we analyse the `ELA-Black-SFW.tif` and `ELA-White-SFW.tif` interactive files: the more we compress, the more they become visible either on the background (White) or on the letters (Black) of the samples.

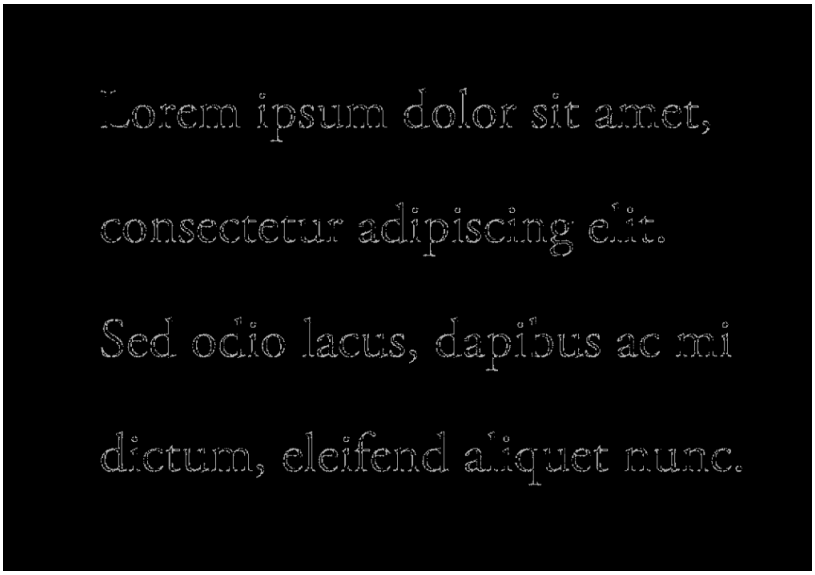

Figure 33: ELA of compression level 0 for Black

Again, the **file sizes** are proportional to the selected compression quality: the higher the quality the bigger the corresponding file size of the resulting sample,

with a file size gap between compression qualities 51 and 50. The file sizes of the original samples (37.8MB in both cases) is also dramatically reduced.

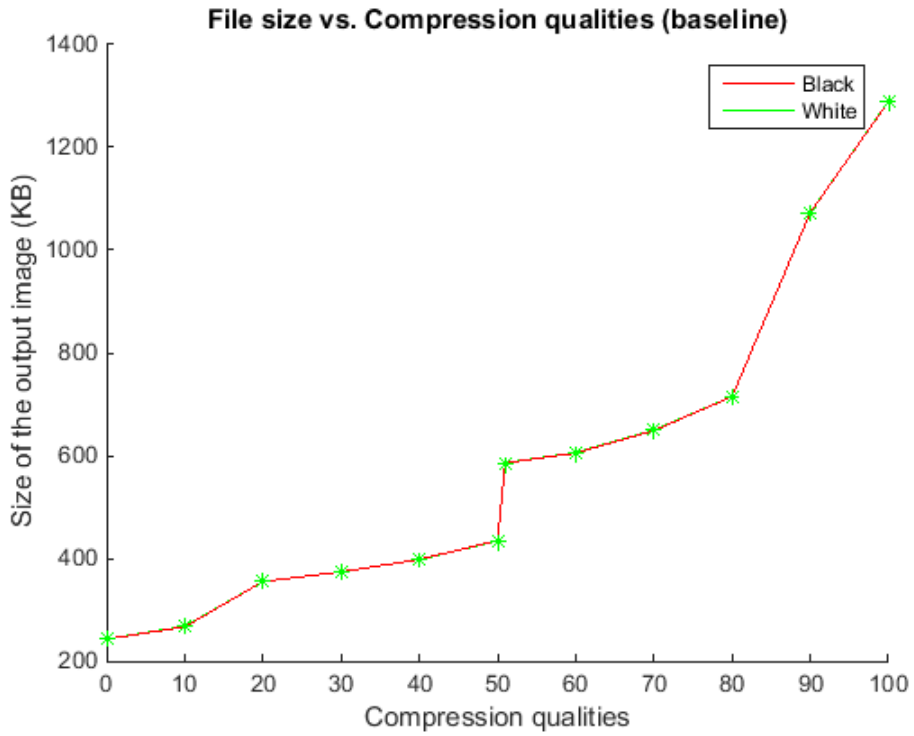

Figure 34: Compression qualities vs. file size. The data set is available as a stand-alone file called `TextDataSet - SaveForWeb.txt`

3.5 Photoshop CS5 `Save as`: results

Interestingly, Photoshop CS5 suggests grouping the available `Save as` compression levels into four different sets:

- *Low quality* for compression levels 0-4;
- *Medium quality* for compression levels 5-7;
- *High quality* for compression levels 8-9;
- *Maximum quality* for compression levels 10-12.

In the previous pages, however, we have noticed that there is usually a **considerable size gap** between compression levels 12 and 11, despite producing samples with very similar features. In fact, images compressed at level 12 are generally the best looking ones, showing sharp details, realistic tones and no artefacts; samples compressed at level 11, instead, might display some artefacts in certain critical situations (edges, smooth areas, ...). To understand these differences we can use `JPEGsnoop` and analyse the data stream of the samples[4]. Consider for example the geometrical image GNS2C; as the marker `DQT` shows, the luminance quantization table used for compression level 12 is the following:

```
DQT, Row #0:   1   1   1   1   1   1   1   2
DQT, Row #1:   1   1   1   1   1   1   1   2
DQT, Row #2:   1   1   1   1   1   1   2   2
DQT, Row #3:   1   1   1   1   1   2   2   3
DQT, Row #4:   1   1   1   1   2   2   3   3
DQT, Row #5:   1   1   1   2   2   3   3   3
DQT, Row #6:   1   1   2   2   3   3   3   3
DQT, Row #7:   2   2   2   3   3   3   3   3
```

The chrominance quantization table used for this compression level is instead:

```
DQT, Row #0:   1   1   1   2   3   3   3   3
DQT, Row #1:   1   1   1   2   3   3   3   3
DQT, Row #2:   1   1   2   3   3   3   3   3
DQT, Row #3:   2   2   3   3   3   3   3   3
DQT, Row #4:   3   3   3   3   3   3   3   3
DQT, Row #5:   3   3   3   3   3   3   3   3
DQT, Row #6:   3   3   3   3   3   3   3   3
DQT, Row #7:   3   3   3   3   3   3   3   3
```

Both tables have very low quantizing factors, which means that high-frequency details will be rarely rounded to zeros thus preserving even those details that the human eye cannot perceive. This also explains why level 12 samples have the

[4]The whole transcription of quantization tables, Huffman tables and chroma subsampling used by Photoshop CS5 `Save as` is available in appendix 1.

biggest file sizes and can be easily compared to their high-definition counterparts. The luminance quantization table used by compression level 11 is instead the following:

```
DQT, Row #0:   1   1   1   2   3   3   4   5
DQT, Row #1:   1   1   1   2   3   4   4   6
DQT, Row #2:   1   1   2   3   4   4   5   7
DQT, Row #3:   2   2   3   4   4   5   7   8
DQT, Row #4:   3   3   4   4   5   7   8   8
DQT, Row #5:   3   4   4   5   7   8   8   8
DQT, Row #6:   4   4   5   7   8   8   8   8
DQT, Row #7:   5   6   7   8   8   8   8   8
```

The chrominance quantization table is:

```
DQT, Row #0:   1   2   4   7   8   8   8   8
DQT, Row #1:   2   2   4   7   8   8   8   8
DQT, Row #2:   4   4   7   8   8   8   8   8
DQT, Row #3:   7   7   8   8   8   8   8   8
DQT, Row #4:   8   8   8   8   8   8   8   8
DQT, Row #5:   8   8   8   8   8   8   8   8
DQT, Row #6:   8   8   8   8   8   8   8   8
DQT, Row #7:   8   8   8   8   8   8   8   8
```

Despite being still very low, the quantizing factors of these tables are higher than those used by top-quality compression level 12, and therefore more details are lost during compression. `JPEGsnoop` also shows that both compression levels do not perform chroma subsampling...

```
Component[1]: ... (Subsamp 1 x 1) ... (Lum: Y)
Component[2]: ... (Subsamp 1 x 1) ... (Chrom: Cb)
Component[3]: ... (Subsamp 1 x 1) ... (Chrom: Cr)
```

...and that despite having the same amount of code words, the Huffman tables used by compression levels 12 and 11 are *not* the same. In fact, Photoshop CS5 selects **different sets of Huffman tables depending on the compression level**: the

more we compress an image, the fewer code words are used for the encoding of the AC components. Therefore it appears that the differences between compression levels 12 and 11 can be explained as **a matter of quantization and Huffman tables**.

Compression	DC-Luma	DC-Chroma	AC-Luma	AC-Chroma
12	12	12	162	162
11	12	12	162	162
10,9,8,7,6	12	12	162	162
5,4,3	12	12	114	111
2,1,0	12	12	98	91

Table 2: Number of code words for each Huffman table

Is it possible to extend these observations to the remaining compression levels offered by Photoshop CS5 `Save as`? Indeed, it is true that quantization tables with higher coefficients and Huffman tables with fewer code words can explain at least partially the visual defects showed by most low-end compression levels; still, to get the whole picture we must also consider **how chroma subsampling impacts the overall quality of the compressed samples**. To this end, let's focus on the remarkable size gap between compression levels 7 and 6. As before, consider the geometrical image GNS2C; the luminance and chrominance quantization tables used by compression level 7 are:

```
DQT, Row #0:  10   7  11  18  22  27  34  17
DQT, Row #1:   7   8  10  17  23  23  12  12
DQT, Row #2:  11  10  14  22  23  12  12  12
DQT, Row #3:  18  17  22  23  12  12  12  12
DQT, Row #4:  22  23  23  12  12  12  12  12
DQT, Row #5:  27  23  12  12  12  12  12  12
DQT, Row #6:  34  12  12  12  12  12  12  12
DQT, Row #7:  17  12  12  12  12  12  12  12

DQT, Row #0:  11  14  31  34  20  20  17  17
```

```
DQT, Row #1:  14  19  24  14  14  12  12  12
DQT, Row #2:  31  24  14  14  12  12  12  12
DQT, Row #3:  34  14  14  12  12  12  12  12
DQT, Row #4:  20  14  12  12  12  12  12  12
DQT, Row #5:  20  12  12  12  12  12  12  12
DQT, Row #6:  17  12  12  12  12  12  12  12
DQT, Row #7:  17  12  12  12  12  12  12  12
```

As expected both tables display higher quantizing factors than those used by compression level 8. As before, no chroma subsampling is performed:

```
Component[1]: ... (Subsamp 1 x 1), ... (Lum: Y)
Component[2]: ... (Subsamp 1 x 1), ... (Chrom: Cb)
Component[3]: ... (Subsamp 1 x 1), ... (Chrom: Cr)
```

Compression level 7 and 6 also share the same Huffman tables. Surprisingly, `JPEGsnoop` shows that the luminance and chrominance quantization tables used by compression level 6 have **coefficients whose values are lower than those used by compression level 7**:

```
DQT, Row #0:   8   6   9  14  17  21  28  17
DQT, Row #1:   6   6   8  13  18  23  12  12
DQT, Row #2:   9   8  11  17  23  12  12  12
DQT, Row #3:  14  13  17  23  12  12  12  12
DQT, Row #4:  17  18  23  12  12  12  12  12
DQT, Row #5:  21  23  12  12  12  12  12  12
DQT, Row #6:  28  12  12  12  12  12  12  12
DQT, Row #7:  17  12  12  12  12  12  12  12

DQT, Row #0:   9   9  11  18  20  20  17  17
DQT, Row #1:   9  10  11  14  14  12  12  12
DQT, Row #2:  11  11  14  14  12  12  12  12
DQT, Row #3:  18  14  14  12  12  12  12  12
DQT, Row #4:  20  14  12  12  12  12  12  12
DQT, Row #5:  20  12  12  12  12  12  12  12
```

```
DQT, Row #6:  17  12  12  12  12  12  12  12
DQT, Row #7:  17  12  12  12  12  12  12  12
```

The reason why compression level 6 still yields smaller files lies on the chroma subsampling performed by this compression level, which is reported below:

```
Component[1]: ... (Subsamp 1 x 1), ... (Lum: Y)
Component[2]: ... (Subsamp 2 x 2), ... (Chrom: Cb)
Component[3]: ... (Subsamp 2 x 2), ... (Chrom: Cr)
```

It appears that compression levels 7 and 6 apply **two different compression strategies**: since 7 does not perform chroma subsampling, to reduce the otherwise huge gap in file size compression level 7 uses two poorer luminance and chrominance quantization tables. Therefore the **image quality of level 7 is actually worse than compression level 6** from a luminance point of view [PP 2011]: unexpectedly, block artefacts are very visible and edges are less sharp with compression level 7 than with compression level 6 (fig. 35).

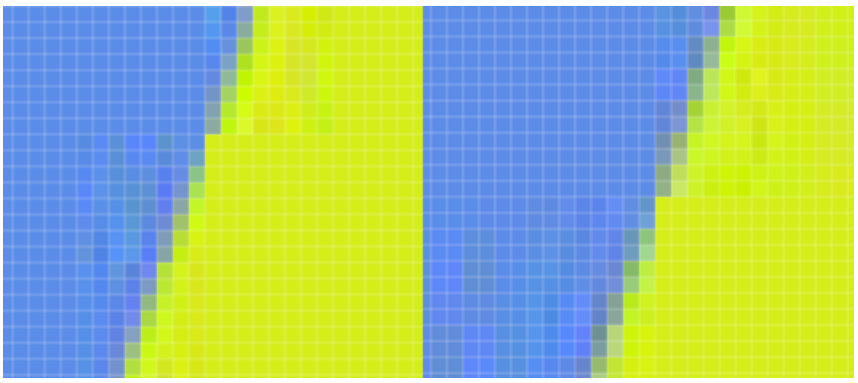

Figure 35: GNS2C baseline "`Save as`": compressions 7 (left) and 6 (right)

To conclude, with the exception of compression level 7 and 6 Photoshop CS5 `Save as` function works as follows:

- The lower the compression level, the higher the coefficients of the corresponding quantization tables.
- The lower the compression level, the fewer code words are listed on the corresponding Huffman tables;

- For compression levels between 6 and 0 chroma subsampling 2×2 is also applied.

All these factors explain why block artefacts, colour bands and discolourations are more and more noticeable the more the original image is compressed.

3.5.1 Polygons with shadows: results

We hve seen in section 3.1.3 that there is a strong relationship between GNS2C (geometric picture without shadows) and GS2C (geometric picture with, shadows) file sizes (table 3); in the previous pages we assumed that this relationship was simply caused by the complexity of compressing and rendering shadows. This is confirmed by `JPEGsnoop`: Photoshop CS5 applies **the same luma-chroma quantization tables, chroma subsampling and Huffman tables** seen before — which means that any difference in file size between samples is caused solely by the content of the image itself and not by different optimization approaches.

Format (compression level)	File size (shadows)	File size (normal)
TIFF (original image)	37.1MB	37.1MB
JPEG (12)	1212KB	1041KB
JPEG (11)	886KB	835KB
JPEG (10)	528KB	495KB
JPEG (9)	488KB	473KB
JPEG (8)	467KB	455KB
JPEG (7)	442KB	439KB
JPEG (6)	289KB	265KB
JPEG (5)	264KB	263KB
JPEG (4)	256KB	256KB
JPEG (3)	250KB	252KB
JPEG (2)	177KB	181KB
JPEG (1)	170KB	175KB
JPEG (0)	167KB	173KB

Table 3: GNS2C vs GS2C JPEG file sizes

3.6 Photoshop CS5 `Save for web` function: results

As before, Photoshop CS5 clusters the available `Save for web` compression qualities into five different sets:

- *Maximum quality* for compression quality 100;
- *Very high quality* for compression qualities 99-80;
- *High quality* for compression qualities 79-60;
- *Medium quality* for compression qualities 59-30;
- *Low quality* for compression qualities 29-0.

Indeed, in the previous pages we have seen that compression quality 100 stands apart from the other compressions, yielding the best-looking results; in fact, its samples are so remarkable that they can easily be compared with their high-definition counterparts showing neat edges, very detailed areas and realistic colours. These observations are supported by the analysis done using `JPEGsnoop`[5]. Consider for example the geometrical image GNS2C; the marker `DQT` shows that this compression quality uses the following luminance quantization table:

```
DQT, Row #0:   1   1   1   1   1   1   1   1
DQT, Row #1:   1   1   1   1   1   1   1   1
DQT, Row #2:   1   1   1   1   1   1   1   2
DQT, Row #3:   1   1   1   1   1   1   2   2
DQT, Row #4:   1   1   1   1   1   2   2   3
DQT, Row #5:   1   1   1   1   2   2   3   3
DQT, Row #6:   1   1   1   2   2   3   3   3
DQT, Row #7:   1   1   2   2   3   3   3   3
```

The chrominance quantization table is instead:

[5]The whole transcription of quantization tables, Huffman tables and chroma subsampling used by Photoshop CS5 `Save for Web` is available in appendix 2.

```
DQT, Row #0:   1   1   1   2   2   3   3   3
DQT, Row #1:   1   1   1   2   3   3   3   3
DQT, Row #2:   1   1   1   3   3   3   3   3
DQT, Row #3:   2   2   3   3   3   3   3   3
DQT, Row #4:   2   3   3   3   3   3   3   3
DQT, Row #5:   3   3   3   3   3   3   3   3
DQT, Row #6:   3   3   3   3   3   3   3   3
DQT, Row #7:   3   3   3   3   3   3   3   3
```

Both tables have very low quantizing factors, which means that **most high-frequency details of the original high-definition picture are preserved** in the compressed samples. `JPEGsnoop` also shows that no chroma subsampling is performed...

```
Component[1]: ... (Subsamp 1 x 1), ... (Lum: Y)
Component[2]: ... (Subsamp 1 x 1), ... (Chrom: Cb)
Component[3]: ... (Subsamp 1 x 1), ... (Chrom: Cr)
```

...and that Photoshop CS5 uses a set of Huffman tables with the standard amount of code words. All these factors explain why compression level 100 returns the best samples and must be regarded as the best compression quality offered by `Save for Web`. We might now ask ourselves: is it safe to assume that the lower the compression quality, the higher the coefficients of the quantization tables and the fewer the code words listed into the Huffman tables? In other words, can we explain the visual defects (block artefacts, colour bands, posterized areas, ...) showed by most low-quality compression samples as **a matter of quantization and Huffman tables alone**? Indeed, scanning each sample with `JPEGsnoop` reveals that `Save for Web` usually follows these principles: the quantizing factors used compression quality 70 are higher than those used by compression quality 80, the quantizing factors used by compression quality 60 are higher than those used by compression quality 70 and so on. As for the Huffman tables, it is true that the more we compress the fewer code words are listed; in fact, Photoshop `Save for Web` uses a different set of Huffman tables depending on the compression level: the more we compress an image, the fewer code words are used for the encoding of both the DC and AC components.

Compression	DC-Luma	DC-Chroma	AC-Luma	AC-Chroma
100-99	12	12	162	162
98-87	12	12	162	162
86-41	12	12	162	162
40-11	12	12	114	111
10-0	12	12	98	91

Table 4: Number of code words for each Huffman table

Notice that `Save for Web` uses the same sets of Huffman tables used by `Save as`, depending of course on the selected compression quality. Still, in the previous pages we noticed a peculiar size gap between compression qualities 51 and 50, which is caused by an additional factor: **chroma subsampling**. `JPEGsnoop` shows that no chroma subsampling is performed by compression quality 51:

```
Component[1]: ... (Subsamp 1 x 1), ... (Lum: Y)
Component[2]: ... (Subsamp 1 x 1), ... (Chrom: Cb)
Component[3]: ... (Subsamp 1 x 1), ... (Chrom: Cr)
```

The same does not apply to compression quality 50:

```
Component[1]: ... (Subsamp 1 x 1), ... (Lum: Y)
Component[2]: ... (Subsamp 2 x 2), ... (Chrom: Cb)
Component[3]: ... (Subsamp 2 x 2), ... (Chrom: Cr)
```

Much like `Save as` compression level 6, `Save for Web` compression quality 50 is a turning point as far as chroma subsampling is concerned: `JPEGsnoop` shows that only compression qualities between 50 and 0 use 2×2 chroma subsampling. Interestingly, the analysis of the quantization tables of these compression qualities shows that `Save for Web` uses a slightly different solution to reduce the otherwise huge file size gap between compression qualities 51 and 50. Compression quality 51 uses the following luma-chroma quantization tables:

```
DQT, Row #0:   8   5   5   8  11  13  15  17
DQT, Row #1:   5   6   6   7  10  12  12  15
```

```
DQT, Row #2:   5   6   6   8  12  13  17  23
DQT, Row #3:   8   7   8  13  13  18  23  34
DQT, Row #4:  11  10  12  13  19  25  33  38
DQT, Row #5:  13  12  13  18  25  33  38  38
DQT, Row #6:  15  12  17  23  33  38  38  38
DQT, Row #7:  17  15  23  34  38  38  38  38

DQT, Row #0:   8   9  16  29  32  38  38  38
DQT, Row #1:   9  14  20  26  38  38  38  38
DQT, Row #2:  16  20  21  38  38  38  38  38
DQT, Row #3:  29  26  38  38  38  38  38  38
DQT, Row #4:  32  38  38  38  38  38  38  38
DQT, Row #5:  38  38  38  38  38  38  38  38
DQT, Row #6:  38  38  38  38  38  38  38  38
DQT, Row #7:  38  38  38  38  38  38  38  38
```

Compression quality 50 uses instead:

```
DQT, Row #0:   8   6   6   8  12  14  16  17
DQT, Row #1:   6   6   6   8  10  13  12  15
DQT, Row #2:   6   6   7   8  13  14  18  24
DQT, Row #3:   8   8   8  14  13  19  24  35
DQT, Row #4:  12  10  13  13  20  26  34  39
DQT, Row #5:  14  13  14  19  26  34  39  39
DQT, Row #6:  16  12  18  24  34  39  39  39
DQT, Row #7:  17  15  24  35  39  39  39  39

DQT, Row #0:   9   8   9  11  14  17  19  24
DQT, Row #1:   8  10   9  11  14  13  17  22
DQT, Row #2:   9   9  13  14  13  15  23  26
DQT, Row #3:  11  11  14  14  15  20  26  33
DQT, Row #4:  14  14  13  15  20  24  33  39
DQT, Row #5:  17  13  15  20  24  32  39  39
DQT, Row #6:  19  17  23  26  33  39  39  39
DQT, Row #7:  24  22  26  33  39  39  39  39
```

It appears that the luminance quantization tables used by compression level 51 and 50 are more or less the same, with compression quality 50 having slightly higher quantizing factors, exactly as expected. Instead, **the chrominance quantization table used by compression quality 50 seems to be the better one**, since its coefficients are generally lower than those used by compression quality 51. Therefore, compression qualities 51 and 50 use two different compression strategies: since compression quality 51 does not perform chroma subsampling, to reduce the potentially huge gap with compression quality 50, it uses a slightly poorer chrominance quantization table. To conclude, with the exception of compression qualities 51 and 50, Photoshop CS5 `Save for Web` works as follows:

- The lower the compression quality, the higher the coefficients of the corresponding quantization tables;
- The lower the compression quality, the fewer code words used on the Huffman tables;
- For compression qualities between 50 and 0 chroma subsampling 2×2 is also applied.

As before, all these elements explain why block artefacts and other discolourations are more and more visible the more the original high-definition image is compressed.

3.7 Photoshop CS5 `Save as` versus `Save for web`

Can we find correspondences between `Save as` (`Sa`) and `Save for Web` (`SfW`)? To answer this question we need to look at the quantization tables, Huffman tables and chroma subsampling techniques used by both compression approaches. Visually, it is difficult to tell `SfW` compression quality 100 from `Sa` compression level 12: both solutions produce remarkably good samples with sharp colours, realistic tones and detailed elements. Still, the quantizing factors used by `SfW` compression quality 100 are lower than those used by `Sa` compression level 12:

```
Luminance quantization table for Save for Web compression level 100:
    DQT, Row #0:   1   1   1   1   1   1   1   1
```

```
    DQT, Row #1:   1   1   1   1   1   1   1   1
    DQT, Row #2:   1   1   1   1   1   1   1   2
    DQT, Row #3:   1   1   1   1   1   1   2   2
    DQT, Row #4:   1   1   1   1   1   2   2   3
    DQT, Row #5:   1   1   1   1   2   2   3   3
    DQT, Row #6:   1   1   1   2   2   3   3   3
    DQT, Row #7:   1   1   2   2   3   3   3   3

Luminance quantization table for Save as compression level 12:
    DQT, Row #0:   1   1   1   1   1   1   1   2
    DQT, Row #1:   1   1   1   1   1   1   1   2
    DQT, Row #2:   1   1   1   1   1   1   2   2
    DQT, Row #3:   1   1   1   1   1   2   2   3
    DQT, Row #4:   1   1   1   1   2   2   3   3
    DQT, Row #5:   1   1   1   2   2   3   3   3
    DQT, Row #6:   1   1   2   2   3   3   3   3
    DQT, Row #7:   2   2   2   3   3   3   3   3

Chrominance quantization table for Save for Web compression level 100:
    DQT, Row #0:   1   1   1   2   2   3   3   3
    DQT, Row #1:   1   1   1   2   3   3   3   3
    DQT, Row #2:   1   1   1   3   3   3   3   3
    DQT, Row #3:   2   2   3   3   3   3   3   3
    DQT, Row #4:   2   3   3   3   3   3   3   3
    DQT, Row #5:   3   3   3   3   3   3   3   3
    DQT, Row #6:   3   3   3   3   3   3   3   3
    DQT, Row #7:   3   3   3   3   3   3   3   3

Chrominance quantization table for Save as compression level 12:
    DQT, Row #0:   1   1   1   2   3   3   3   3
    DQT, Row #1:   1   1   1   2   3   3   3   3
    DQT, Row #2:   1   1   2   3   3   3   3   3
    DQT, Row #3:   2   2   3   3   3   3   3   3
    DQT, Row #4:   3   3   3   3   3   3   3   3
```

```
    DQT, Row #5:   3   3   3   3   3   3   3   3
    DQT, Row #6:   3   3   3   3   3   3   3   3
    DQT, Row #7:   3   3   3   3   3   3   3   3
```

Given that in both cases Photoshop CS5 uses the same set of Huffman tables and performs no chroma subsampling, `SfW` compression quality 100 seems to be a better solution than `Sa` compression level 12 if we want to achieve the best-looking samples. The same considerations applies for `SfW` compression qualities 90, 80, 70 and 60: despite being visually very similar to `Sa` compression levels 11, 10, 9 and 8 respectively, they use better quantization tables with lower coefficients, as we can see in the example below.

```
Luminance quantization table for Save for Web compression quality 80:
    DQT, Row #0:   2   2   2   2   3   4   5   6
    DQT, Row #1:   2   2   2   2   3   4   5   6
    DQT, Row #2:   2   2   2   2   4   5   7   9
    DQT, Row #3:   2   2   2   4   5   7   9  12
    DQT, Row #4:   3   3   4   5   8  10  12  12
    DQT, Row #5:   4   4   5   7  10  12  12  12
    DQT, Row #6:   5   5   7   9  12  12  12  12
    DQT, Row #7:   6   6   9  12  12  12  12  12

Luminance quantization table for Save for Web compression level 9:
    DQT, Row #0:   4   3   4   7   9  11  14  17
    DQT, Row #1:   3   3   4   7   9  12  12  12
    DQT, Row #2:   4   4   5   9  12  12  12  12
    DQT, Row #3:   7   7   9  12  12  12  12  12
    DQT, Row #4:   9   9  12  12  12  12  12  12
    DQT, Row #5:  11  12  12  12  12  12  12  12
    DQT, Row #6:  14  12  12  12  12  12  12  12
    DQT, Row #7:  17  12  12  12  12  12  12  12

Chrominance quantization table for Save for Web compression quality 80:
    DQT, Row #0:   3   3   5   9  13  15  15  15
    DQT, Row #1:   3   4   6  11  14  12  12  12
```

```
    DQT, Row #2:   5   6   9  14  12  12  12  12
    DQT, Row #3:   9  11  14  12  12  12  12  12
    DQT, Row #4:  13  14  12  12  12  12  12  12
    DQT, Row #5:  15  12  12  12  12  12  12  12
    DQT, Row #6:  15  12  12  12  12  12  12  12
    DQT, Row #7:  15  12  12  12  12  12  12  12

Chrominance quantization table for Save for Web compression level 9:
    DQT, Row #0:   4   6  12  22  20  20  17  17
    DQT, Row #1:   6   8  12  14  14  12  12  12
    DQT, Row #2:  12  12  14  14  12  12  12  12
    DQT, Row #3:  22  14  14  12  12  12  12  12
    DQT, Row #4:  20  14  12  12  12  12  12  12
    DQT, Row #5:  20  12  12  12  12  12  12  12
    DQT, Row #6:  17  12  12  12  12  12  12  12
    DQT, Row #7:  17  12  12  12  12  12  12  12
```

`SfW` compression qualities 51 (no chroma subsampling) and 50 (2×2 chroma subsampling) seems to behave better than `Sa` compression levels 7, 6 and 5 (with 7 better than 6, as we have noticed before). This tendency is reversed for the next compressions: `SfW` 40, 30, 20, 10 and 0 all have quantizations tables whose coefficients are consistently higher than those of `Sa` compression levels 4, 3, 2, 1 and 0 respectively[6]. To conclude, it appears that Photoshop `Save as` and `Save for web` settings are **strongly related**: while the quantization tables are usually slightly different, the compression strategies used by these approaches are the same. As an example, we can get an **overview of how much the coefficients of each quantization table grow** when we decrease the compression quality; we analyse the first row of each luminance and chrominance quantization table for both `Save as` and `Save for web` and see that the result is always the same: the higher the compression rate the higher the coefficients, especially for the first five elements of the first row of the quantization tables. However, `Save as` quantization tables tend to have the highest coefficients near the DC components while `Save for web`

[6]See appendices 1 and 2 for all the other quantization tables and chroma subsampling settings used for these comparisons.

coefficients have very high AC components[7]. To sum up, what to do when we want to compress a JPEG file with Photoshop CS5?

- To get the best possible results, use `Save for web 100`; while `Save as 12` is just as good in most circumnstances, `Save for web`'s quantization tables are better from a luminance and chrominance point of view.
- `Save for web 90, 80, 70, 60` are generally better than `Save as 11, 10, 9, 8`, for exactly the same reasons as before;
- Paradoxically, `Save for web 51` > `Save for web 50` > `Save for web 6` > `Save for web 7`.
- `Save as 5, 4, 3, 2, 1, 0` work better than `Save for web 40, 30, 20, 10, 0`.

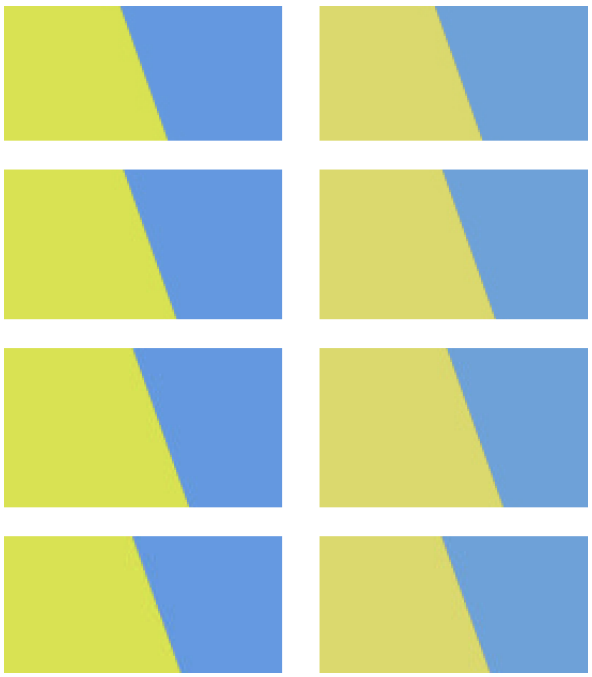

Figure 36: GNS2C baseline: `Save as` (left) 12, 11, 10, 9 vs. `Save for web` (right) 100, 90, 80, 70. Notice the slightly different green colour in the samples on the right: this is still a matter of debate but it appears that this is due to the colour profile conversion (to sRGB) performed by `Save for web` by default.

[7] See `Materials/Exploration in HD/QuantizationTables.png` for further details.

3.8 Additional case study: optimized baseline and progressive compressions

So far we have analysed non-optimized JPEG pictures, obtained using Photoshop CS5 either with `Save as` or `Save for web`. Optimization, however, is offered — sometimes heavily suggested, since it is **the default option for** `Save for web` — by both approaches. But what does it mean to produce an optimized JPEG image? Photoshop documentation suggests that this feature

> (...) *creates an enhanced JPEG with a slightly smaller file size. The Optimized JPEG format is recommended for maximum file compression* [IA-OP 2015].

Indeed, file size is very much reduced when the `optimized` option is ticked, as we can see from the following comparison produced for the geometrical GNS2C image:

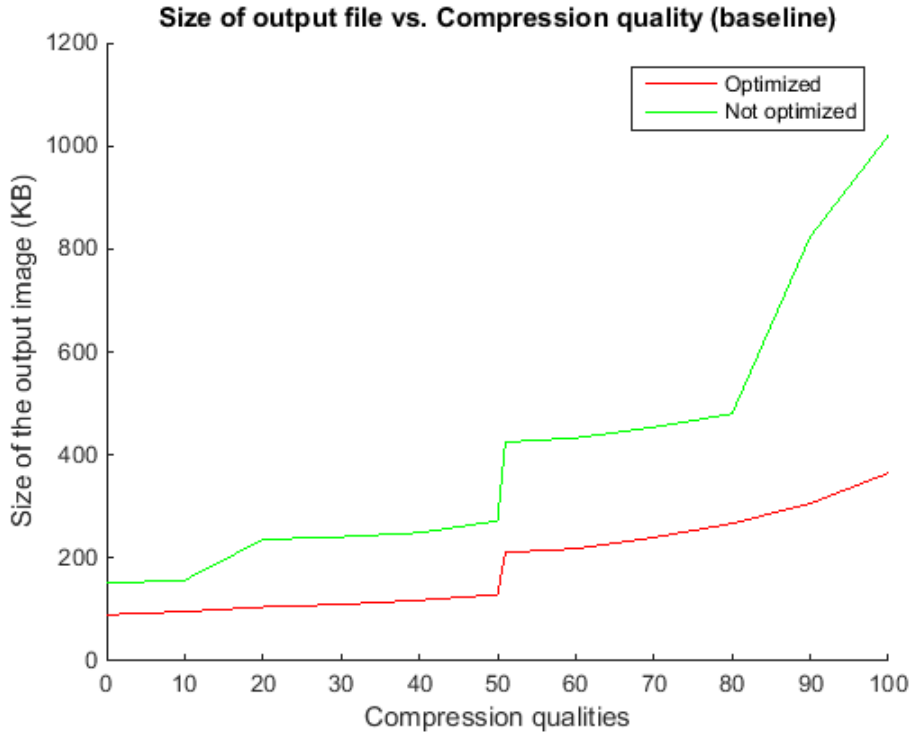

Figure 37: Optimized vs. non-optimized file size (`Save for web`). The data set is available as a stand-alone file called `GeometricDataSet - SaveForWeb.txt`

A quick look at the data provided by `JPEGsnoop` shows that optimized and non-optimized pictures with the same compression level s**hare quantization tables and chroma subsampling settings**; optimized JPEGs, however, use **custom Huffman tables** with a reduced number of code words. In fact, these Huffman tables contain **only the code words that are actually needed to encode the picture** we are working on, which reduces the file size[8]. As for the *progressive optimized* compression offered by both `Save as` and `Save for web`, using `JPEGsnoop` we can see that *progressive optimized* JPEGs share the same quantization tables and chroma subsampling settings with their baseline optimized counterparts; they have, instead, a bigger number of Huffman tables, which are used depending on the scan[9]. Note that despite having more Huffman tables, their content is very limited (fewer code words) — a fact that explains why progressive optimized files have very small sizes, smaller than baseline optimized pictures.

To sum up, Photoshop CS5 JPEG compression highly reduces the size of the original high-definition images, exactly as expected; while it seems to perform best with pictorial images, it shows all its shortcomings when dealing with geometrical, gradient and textual samples, especially when the selected compression rate is very high.

4 Gimp JPEG compression

GIMP (*GNU Image Manipulation Program*) is an open source and free raster graphic editor initially developed in 1995 and now available on Linux, OS X and Microsoft Windows. Exactly as Adobe Photoshop, GIMP offers a fully customizable interface, full-screen mode and extended digital retouching and photo enhancement features; it can also be extended with plug-ins and has a native file format, `XCF`. GIMP offers **only one method** to save files as JPEG images, the `Export` feature, with 101 different compression levels, optimization and progressive options, selectable chroma subsampling settings (4:4:4, horizontal or vertical 4:2:2, 4:2:0) and different DCT coefficients (either integers, fast integers or floating point values).

[8] See appendix 3 for an example.

[9] See appendix 3 for another example.

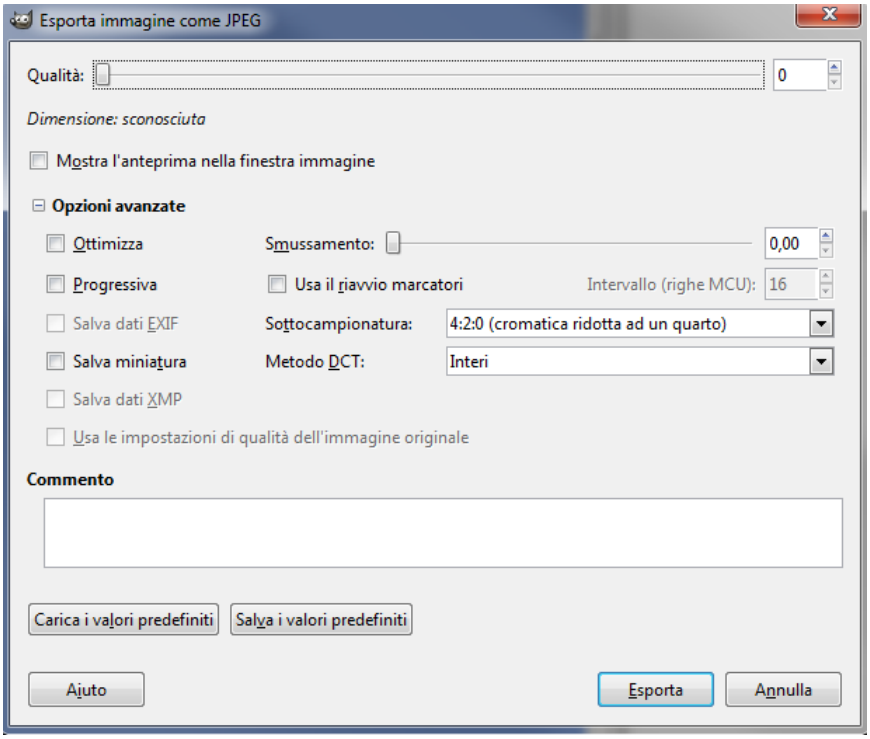

Figure 38: `Export` panel (Italian version)

Just like Adobe Photoshop CS5, GIMP 2 offers numerous options to perform JPEG compression: with so many possible combinations, this tutorial could be endless. Therefore, in the following pages we will focus on compression qualities 100, 90, 80, 70, 60, 50, 40, 30, 20, 10 and 0 with chroma subsampling 4:4:4, 4:2:0 (horizontal and vertical) and 4:2:0 and integer DCT coefficients. We will also consider two additional situations: how different types of DCT coefficients impact the JPEG files produced by GIMP and what changes when the *optimized* and *progressive* options are ticked.

4.1 Geometric images

To proceed with the analysis we save the original geometric images GNS2C and GNS*C twelve times for each possible chroma subsampling setting, with compression qualities 100, 90, 80, 70, 60, 50, 40, 30, 20, 10 and 0. Then, using an image editor of our choice, we examine each sample zooming at 2000% and 200%. We obtain the following results:

GNS2C.

- **Chroma subsampling 4:4:4**. With this setting we get the best-looking results, exactly as expected. At compression quality 100 edges are very sharp, well-defined and show no visible artefacts: the result is so remarkable that, if compared with the original high-definition picture, there are no visible differences.

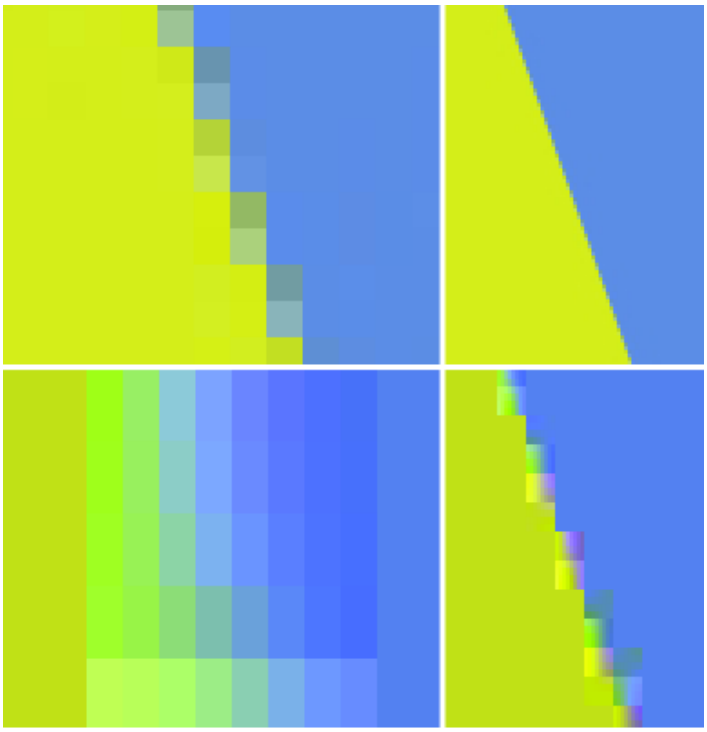

Figure 39: GNS2C 4:4:2 (vertical), compression 100 (top) and 0

Compression qualities 90 and 80 still return very good results: the edges of the polygons still look neat and well-defined, but if we zoom at 2000% we can notice some minor artefacts. As for the other compression qualities, GIMP behaves as expected: the lower the selected compression level the more the overall quality of the picture worsen. The worst results are obtained with compression qualities between 50 and 0 and show very visible block artefacts and discolourations on the edges of the polygons.

- **Chroma subsampling 4:2:2**, horizontal and vertical. GIMP still returns very good samples: when the compression quality is set between 100 and 80 the results are visually on the same level as those obtained with chroma subsampling 4:4:4. The more we compress, however, the more the pictures show block artefacts and discolourations on the edges of the polygons: despite looking acceptable when zooming at 50-100%, the overall result is very unnatural if we zoom further, with extremely visible block artefacts on the edges of the polygons.
- **Chroma subsampling 4:2:0**. Finally, with this setting we have the worst-looking results: block artefacts and discolourations on the edges of the polygons are very visible at compression 80 and below. Still, with compression quality set between 100 and 90 we can obtain good samples, with unnoticeable artefacts and no discolourations.

GNS*C.

- **Chroma subsampling 4:4:4**. With chroma subsampling 4:4:4 we get the best-looking samples. In particular, compression qualities 100 and 90 yield the best results: polygons look very sharp and there are no visible artefacts. The overall quality is so high that a comparison between these samples and their high-definition counterparts show no discernible differences. Compressions between 80 and 50 still look acceptable with only partially visible block artefacts; when we select a compression level lower than 50, however, the samples show a great number of block artefacts, discolourations and blurred areas.
- **Chroma subsampling 4:2:2**, horizontal and vertical. With chroma subsampling 4:2:2 we obtain very good samples, especially when compression quality is set between 100 and 80. The overall quality of the samples plummets, however, when we select lower compression qualities: pictures are still acceptable when zooming at 50-100% but show many block artefacts on the edges of the polygons if we zoom further.
- **Chroma subsampling 4:2:0**. Finally, with chroma subsampling 4:2:0 we have the worst-looking results, since block artefacts on the edges of the polygons become very visible at compression 80 and below. Still,

compression qualities 100 and 90 return good samples with well-defined edges and unnoticeable discolourations.

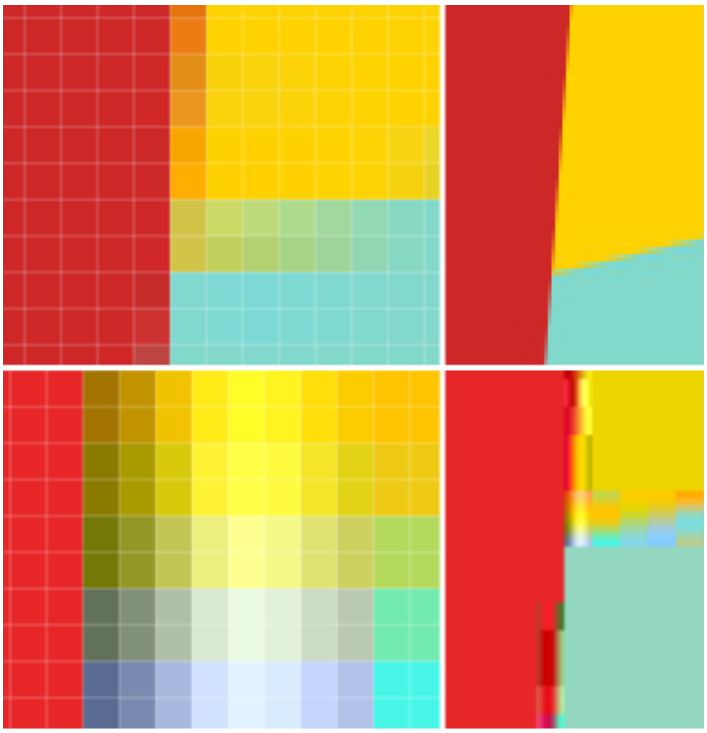

Figure 40: GNS*C 4:4:2 (vertical), compression 100 (top) and 0

Visually, the differences between the various compressions performed by GIMP 2 can be observed if we analyse the `GIMP-ELA-GNS2C.tif` and `GIMP-ELA-GNSC.tif`[10] interactive files: the more we compress, the more the critical areas of the samples (edges, areas where colours abruptly change, ...) show vivid colours and block artefacts (fig. 41). As for the **file sizes**, as expected the more we compress the original picture the smaller the respective sample. Chroma subsampling 4:4:4 always returns the biggest files while 4:2:0 yields the smallest pictures; chroma subsampling settings 4:2:2 horizontal and 4:2:2 vertical are instead extremely similar and a good compromise between the other two. Finally, no matter the chroma subsampling, the biggest gap in size is always between compression qualities 90 and 100 (fig. 42).

[10]For simplicity's sake, these files consider the 4:4:4 chroma subsampling only.

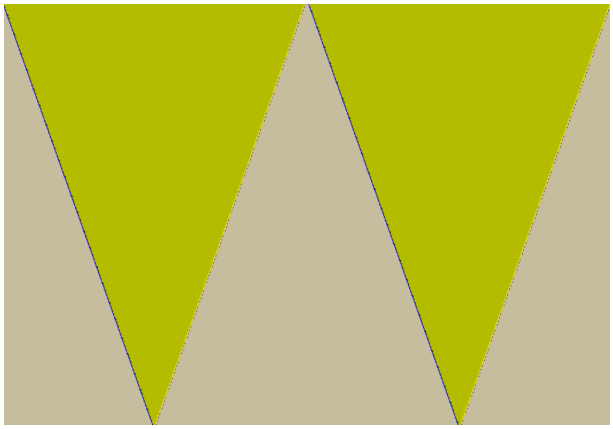

Figure 41: GIMP-ELA-GNS2C.tif with compression quality 0

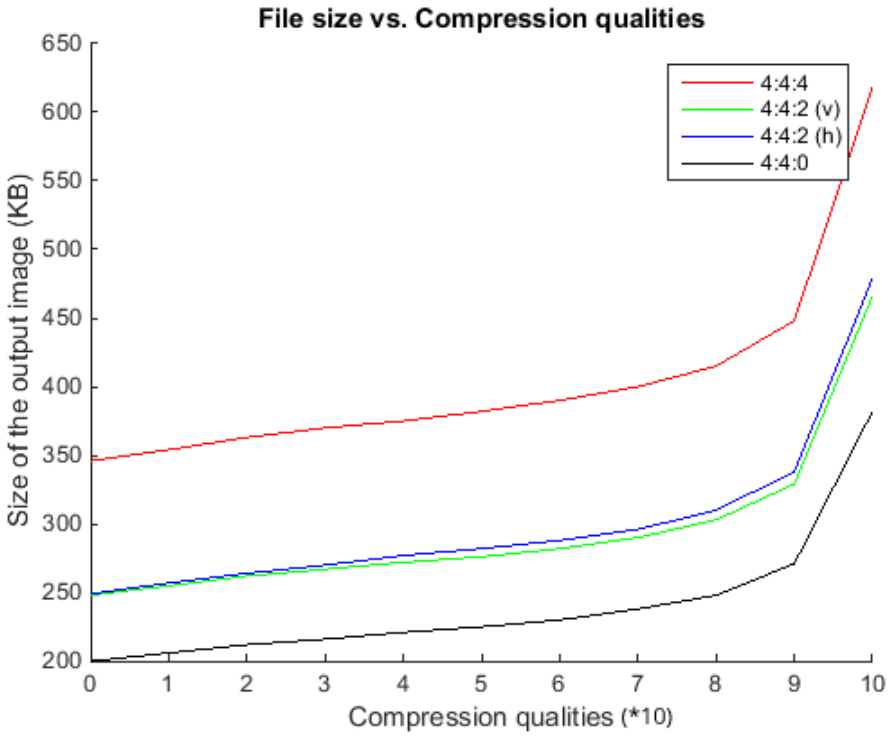

Figure 42: GNS2C file sizes vs. compression. The data set is available as `GeometricDataSet - GNS2C.txt`

4.1.1 Additional case study: geometric image with shadows

In the previous pages we have analysed how GIMP 2 compresses pictures with simple polygons. It may be interesting, however, to see what happens when we add some shadows to the polygons, like in the following example:

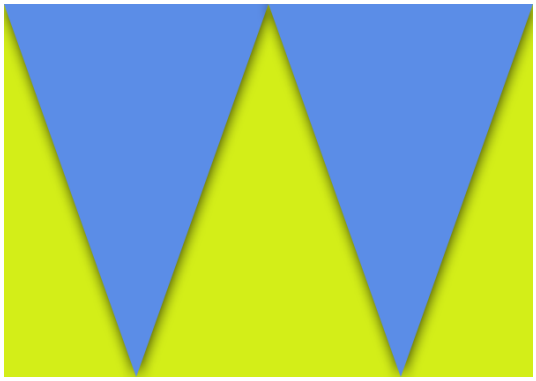

Figure 43: GS2C with sharp edges and shadows

Let's save 11 different versions of `GS2C`, with compression levels varying from 0 to 100, for each available chroma subsampling setting. Then, using an image editor of our choice, we examine each sample zooming at 2000% and 200%.

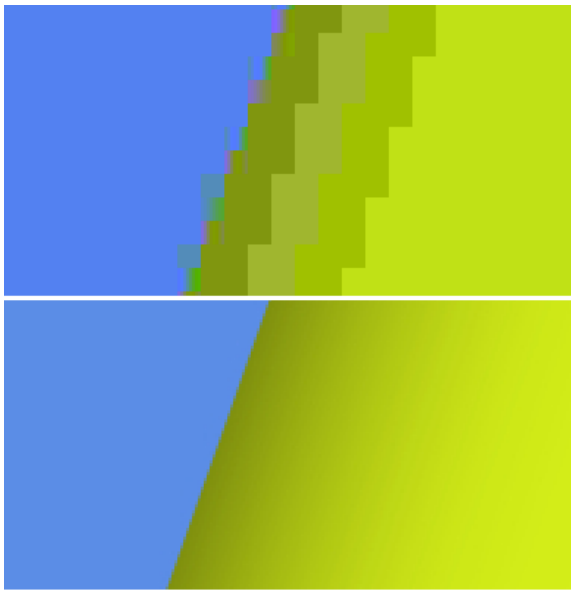

Figure 44: GS2C compressed at qualities 0 (top) and 100 (4:4:4)

As expected, **shadows are critical** when compressing: while they appear smooth and compact when high compression qualities are selected (100, 90), they show very visible block artefacts when lower compression qualities are chosen. As for the **file size**, there is a close relationship between GNS2C and GS2C samples with the same compression qualities: when the compression level is high enough samples with shadows are bigger in size, otherwise the size is more or less the same, as the following plot shows:

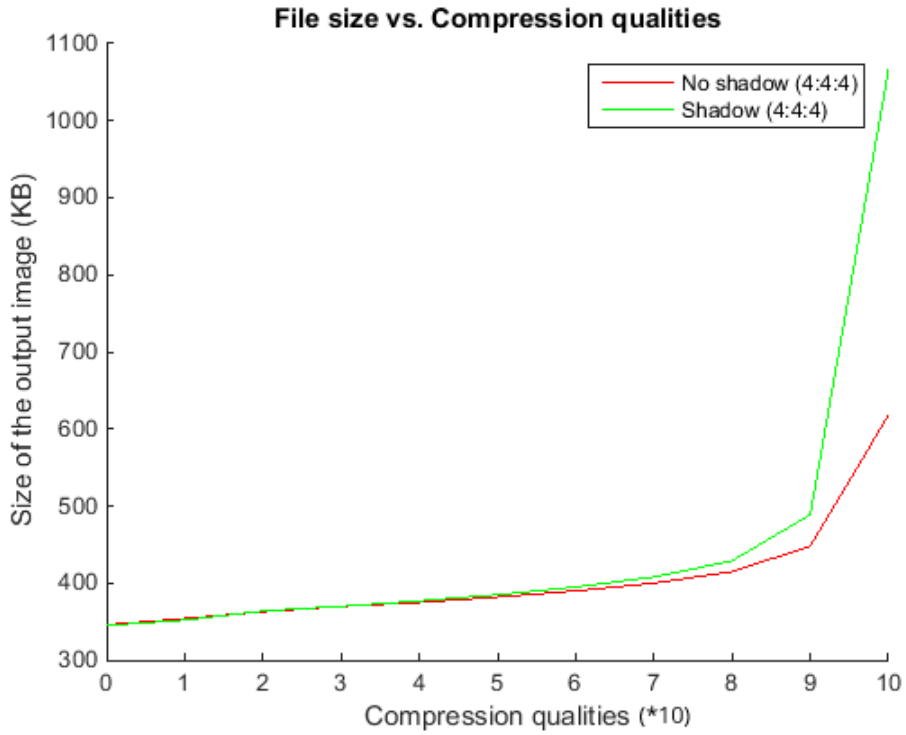

Figure 45: GS2C vs. GNS2C file sizes. The data set is available as a stand-alone file called `GeometricDataSet - GNS2C.txt`

Furthermore, the biggest gap in file size is between compression qualities 90 and 100, exactly as noticed before.

4.2 Pictorial images

As before, we use GIMP 2 `Export` function and obtain 11 different samples for each chroma subsampling setting, varying their compression qualities between 100

and 0. Next, we evaluate each sample using an image editor, zooming at 200% and 2000%. We obtain the following results:

LEP.

- **Chroma subsampling 4:4:4**. As noticed before, chroma subsampling 4:4:4 yields the best-looking results. At compression qualities 100 and 90 the samples can be easily compared to the original high-definition picture: critical areas such as the lake, the blue sky and the clouds look very smooth and detailed, with no visible artefacts or discolourations. Compression qualities 80 and 70 still return acceptable samples since small artefacts are visible mainly on the sky. As for the other compression qualities, the lower the selected compression level the more the overall quality of the picture worsen. The worst-looking results are obtained with compression qualities set between 20 and 0. In particular, at compression quality 0 the sample is completely posterized and, as such, practically useless.
- **Chroma subsampling 4:2:2**, horizontal and vertical. With chroma subsampling 4:2:2 GIMP returns very good samples: when the compression quality is set between 100 and 90 the samples are so detailed and sharp that they can easily be compared to their high-definition counterpart. Selecting other compression levels, however, undermines the overall quality of the samples: compressions between 80 and 40 still yield acceptable results especially if we do not zoom past 50-100%; at 200% the samples look very unnatural, with visible block artefacts and textures on the smoother surfaces.
- **Chroma subsampling 4:2:0**. Finally, with chroma subsampling 4:2:0 we have the worst-looking results: block artefacts on the lake, the sky and the mountains are very visible at compression quality 80 and below. Still, compression qualities 100 and 90 yield very good results, with sharp details and no block artefacts. In fact, these samples are so detailed that it is easy to mix up the original high-definition pictures with the compressed images.

Figure 46: LEP compressed at 0 (top), 50 and 100, chroma subsampling 4:4:4

LLP.

- **Chroma subsampling 4:4:4**. As before, chroma subsampling 4:4:4 returns the best-looking samples, with colours smoothly blended together at compression qualities 100, 90. Compression qualities 80, 70, 60 and 50 still return acceptable samples, with mostly unnoticeable artefacts and posterized areas; the remaining compression qualities return instead the worst results, with many block artefacts and posterized areas

on the critical portions of the pictures. In particular, compression qualities between 20 and 0 are practically useless since the corresponding samples have a completely unnatural look.

Figure 47: LLP compressed at 10 (top), 50 and 100 (bottom) quality levels, chroma subsampling 4:2:2 vertical

- **Chroma subsampling 4:2:2**, horizontal and vertical. With chroma subsampling 4:2:2 GIMP 2 return very good samples: when the com-

pression quality is set between 100 and 90 the results are almost identical to the original high-definition image. Selecting other compression levels, however, drastically reduces the quality of the samples: edges and shapes are still distinguishable but there is an unnatural pink hue all over the samples. Block artefacts are also very visible, especially with compression qualities 50-0.

- **Chroma subsampling 4:2:0**. With chroma subsampling 4:2:0 we have the worst-looking results: block artefacts on the trees and the sky are very visible at compression quality 80 and below. In particular, compression qualities between 30 and 0 return very posterized and pixelated samples, which are practically useless in most situations.

Visually, we can see the amount of artefacts produced by each different compression using the `GIMP-ELA-LEP.tif` and `GIMP-ELA-LLP.tif` interactive files: as expected, the higher the compression rate the more the samples differ from their original counterpart, showing posterized areas and block artefacts.

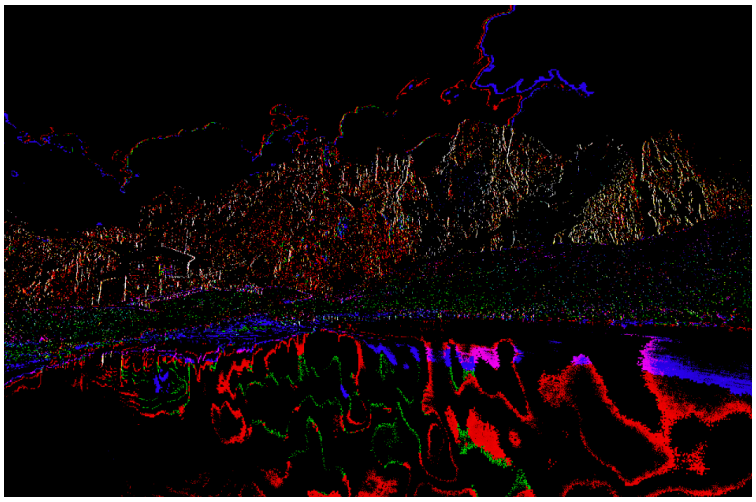

Figure 48: LEP ELA for compression quality 0, chroma subsampling 4:4:4

As for the **file sizes**, unsurprisingly compression reduces the size of the samples. Overall, we obtain the smallest files with chroma subsampling 4:2:0, the biggest files with chroma subsampling 4:4:4. As before, no matter the chroma subsampling, the biggest gap in size is always between compression qualities 90 and 100.

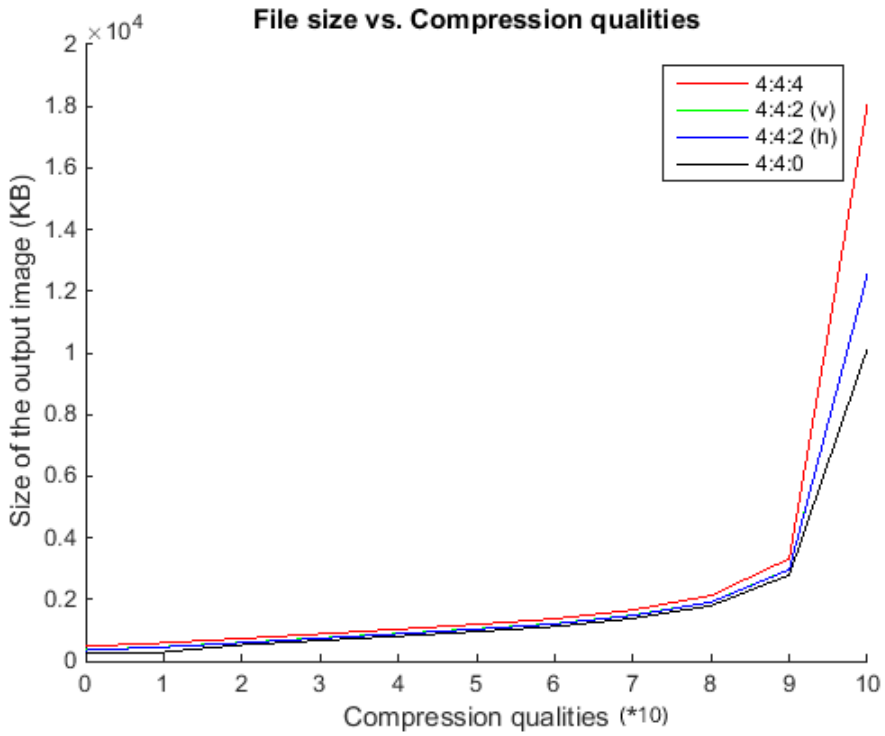

Figure 49: LEP files sizes vs. compression qualities. The data set is available as a stand-alone file called `PictorialDataSet - LEP.txt`

4.3 Solid and gradient images

As before, we use GIMP 2 `Export` feature to obtain 11 different samples for each chroma subsampling setting, varying the selected compression quality between 100 and 0. Next, we analyse the results with an image editor of our choice: by zooming at 2000% and 200% we obtain the results listed below.

S2, S3.

- **Chroma subsampling 4:4:4**. Chroma subsampling 4:4:4 is still the best option to obtain good-looking samples. Compression qualities 100 and 90 yield the best-looking results, with almost no visible bands where different colours are juxtaposed. Compression qualities between 70 and 40 still yield acceptable samples since colour bands are mostly unnoticeable, especially if we do not zoom past 200%. The other compression

qualities show instead all the downsides of performing a JPEG compression on vector images: colour bands are extremely visible.

- **Chroma subsampling 4:2:2**, horizontal and vertical. With chroma subsampling 4:2:2 GIMP 2 return very acceptable images: when the compression quality is set at 100 or 90 the results look sharp and well-detailed, and are almost identical to the original high-definition image. Selecting other compression levels, however, drastically reduces the overall quality of the samples: despite showing distinguishable edges and shapes, colour bands are unfortunately very visible.
- **Chroma subsampling 4:2:0**. Finally, with chroma subsampling 4:2:0 the samples shows visible colour bands at compression quality 70 and below. Still, compression qualities between 100 and 80 produce acceptable samples, with only minor artefacts.

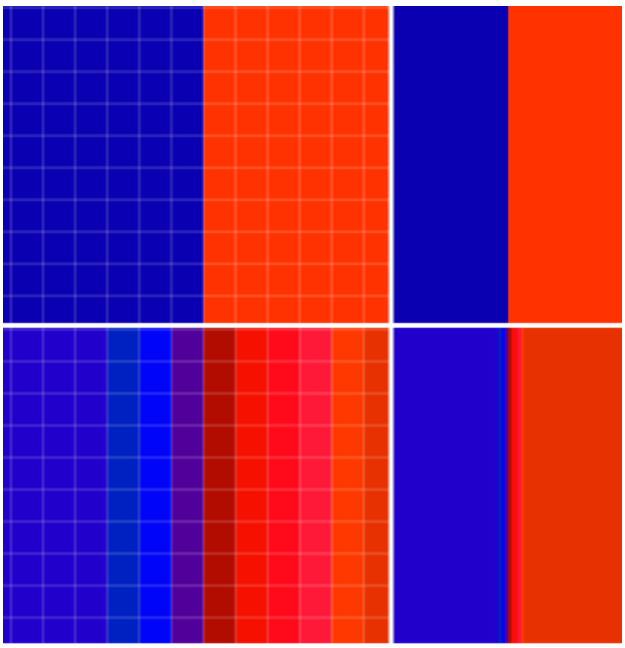

Figure 50: S3 at compression qualities 100 (top) and 0 (bottom), with chroma subsampling 4:4:4

S2S, S3S.

- **Chroma subsampling 4:4:4**. Unsurprisingly, we obtain the best re-

sults with chroma subsampling 4:4:4: compression qualities set at 100, 90 and 80 show perfectly blended colours and no colour bands. Compression levels 70 and 60 still yield acceptable results, with minor colour bands and artefacts; all the other compression qualities are are instead the worst, since colour bands are very noticeable.

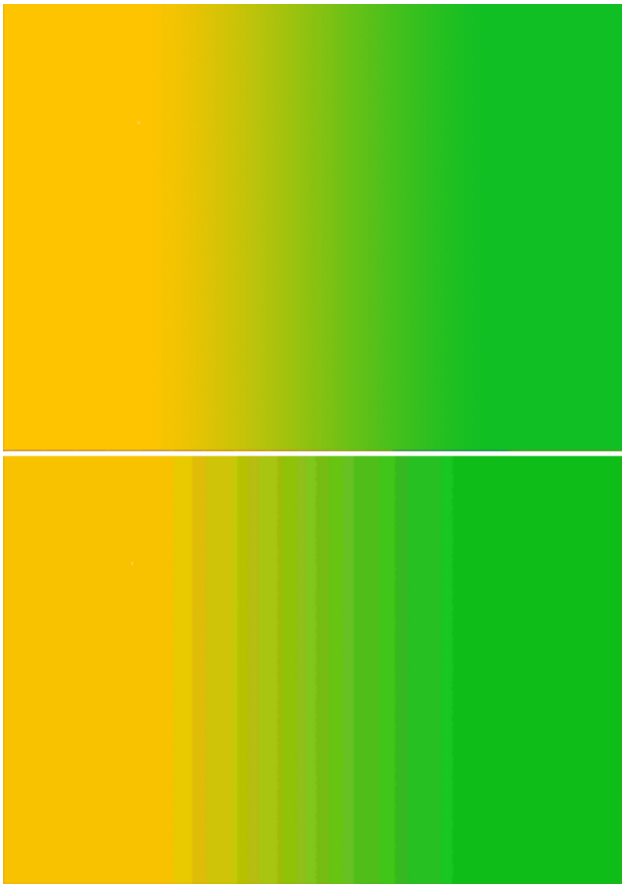

Figure 51: S2S at compression qualities 100 (top) and 0 (bottom), with chroma subsampling 4:4:4

- **Chroma subsampling 4:2:2**, horizontal and vertical. With chroma subsampling 4:2:2 GIMP produces samples that can still be compared with their high-definition counterparts: with compression qualities 100 and 90 colour bands are not visible and colours are perfectly mixed together. The overall quality of the images is nonetheless drastically reduced when other compression qualities are selected, with very vis-

ible colour bands all over the images. Compression qualities 20-0 are especially critical.

- **Chroma subsampling 4:2:0**. Finally, with chroma subsampling 4:2:0 we have the worst-looking results: colour bands are extremely visible at compression 80 and below. Compression 100 and 90 still produce acceptable results but, if we zoom, we can still notice some colour bands.

The same colour bands are visible if we analyse the `GIMP-ELA-S2.tif` and `GIMP-ELA-S2S.tif` interactive files: the more we compress, the more the samples show these discolourations, supporting the idea that the higher the compression rate, the lower the overall quality of the samples. As for the **file size**, the following plot shows the same behaviour of the previous ones:

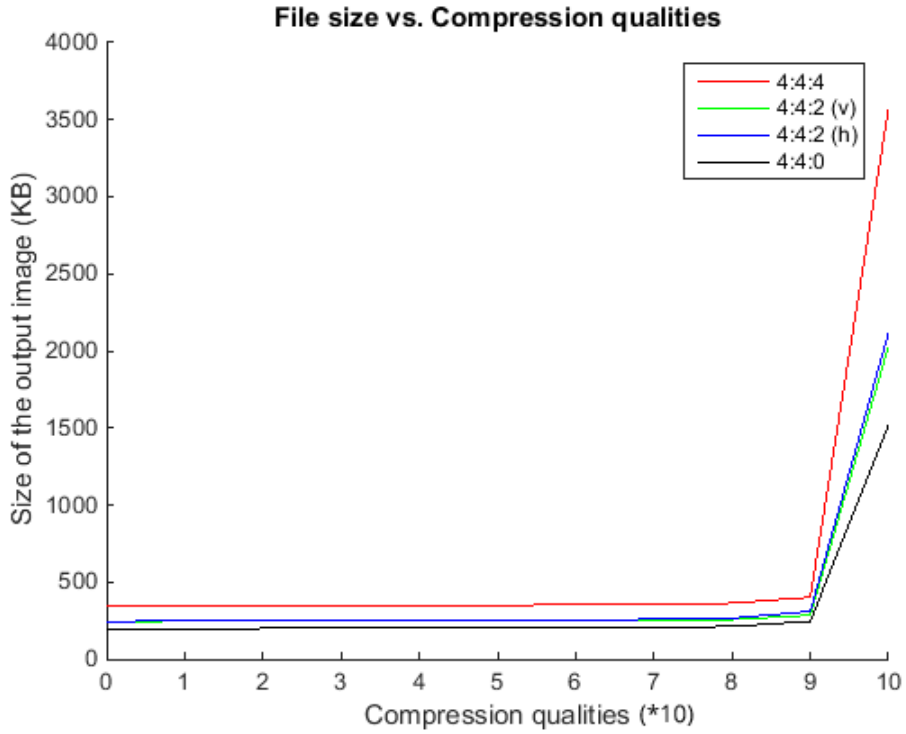

Figure 52: S2 file sizes vs. compression qualities. The data set is available as a stand-alone file called `ColorBlockDataSet - S2.txt`

As we can see, chroma subsampling 4:4:4 returns the biggest files while chroma subsampling 4:2:0 yields the smallest samples. Chroma subsampling 4:2:2 vertical

and horizontal are instead more or less the same. In any case, the biggest file size gap is always between compression qualities 100 and 90; compression quality 100 also returns samples with very different file sizes depending on the selected chroma subsampling.

4.4 Textual images

To analyse the set of textual pictures we use the `Export` feature and obtain 11 different samples for each chroma subsampling setting, varying the selected compression quality between 0 and 100. Next, we analyse the resulting samples with an image editor of our choice, zooming at 200% and 2000%. The results are the following:

Black.

- **Chroma subsampling 4:4:4**. Chroma subsampling 4:4:4 is still the go-to solution to obtain the best possible results: with compression qualities 100, 90 and 80 the samples look sharp, easily readable and show no grey artefacts on the letters of the text. Compression qualities 70, 60 and 50 still return acceptable results, with only minor grey artefacts on the letters. As for the remaining compression qualities, the text is still very much readable but shows imperfections, blurred areas and grey artefacts, exactly as expected.
- **Chroma subsampling 4:2:2**, horizontal and vertical. Chroma subsampling 4:2:2 returns remarkably good results: compression qualities 100, 90 and 80 still produce the best-looking samples, with no gray artefacts. All the other compression qualities show instead all the downsides of performing a JPEG compression on textual images: grey artefacts, discolouration, text is less uniform and letters are not perfectly rounded.
- **Chroma subsampling 4:2:0**. Finally, chroma subsampling 4:2:0 returns the worst-looking files, with many grey artefacts and discolourations on the letters of the text starting from compression quality 80. Compression qualities 100 and 90 yield instead good samples, with no visual artefacts. Note that the text is always easily readable, exactly as before.

White.

- **Chroma subsampling 4:4:4**. As before, chroma subsampling 4:4:4 produces the best-looking results: compression qualities 100, 90 and 80 returns samples with such detailed and contrasted letters that they can easily be compared to their high-definition counterpart. Compression qualities 70, 60 and 50 still return acceptable results, with only minor grey artefacts on the white background. The other compression qualities, instead, yield samples with blurred areas and major grey artefacts. Still, the text is always readable.

Figure 53: `White` compressed at qualities 100 (bottom) and 0, chroma subsampling 4:2:0

- **Chroma subsampling 4:2:2**, horizontal and vertical. Chroma subsampling 4:2:2 produces very good results that can easily be compared to the original high-definition image if we select compression qualities between 100 and 80. Every other compression quality show instead some degree of discolouration: letters are not perfectly rounded and borders have some grey artefactss. Still, the text is perfectly readable

even when selecting the highest compression rates.

- **Chroma subsampling 4:2:0**. With this setting GIMP 2 produces the worst-looking files: the background show grey block artefacts starting from compression quality 80. Compression qualities 100 and 90 produce instead good pictures: the artefacts are unnoticeable and the letters are detailed, with no stained areas. Again, despite the lower quality of the samples, the text is still perfectly readable (fig. 53).

Grey artefacts are visible if we analyse the `GIMP-ELA-Black.tif` and `GIMP-ELA-White.tif` interactive files: the more we compress, the more the samples show these discolourations.

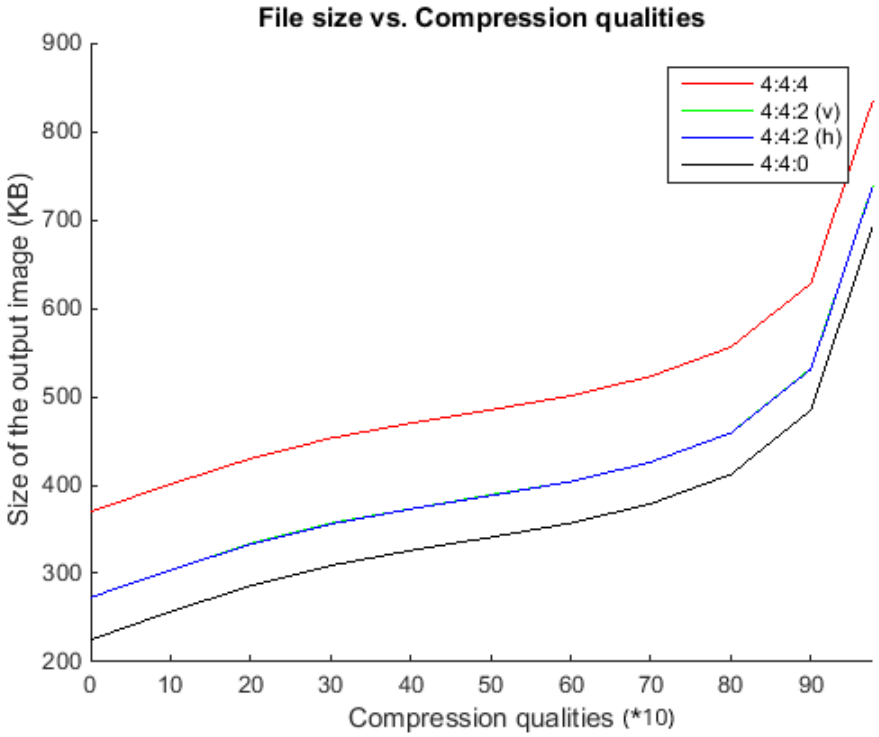

Figure 54: `White` file sizes vs. compression qualities. Note that the green line is covered by the blue one. The data set is available as a stand-alone file called `TextualDataSet - White.txt`

As for the **file sizes**, we can see that chroma subsampling 4:2:2 sits inbetween the remaining settings, producing files with almost identical sizes. As before,

chroma subsampling 4:4:4 returns the biggest files while chroma subsampling 4:2:0 yields the smallest samples. The most remarkable gap in file size is again between compression qualities 100 and 90.

4.5 GIMP JPEG compression: results

So far we have visually analysed the samples produced by GIMP 2 JPEG compression, noting that compression quality 100 usually produces the best possible results: edges are sharp, colours are perfectly blended together, shadows are well-defined, text areas are readable and show no artefacts. Compression quality 90, which should theoretically be closely related to compression quality 100, generally returns visually appealing samples whose size is, interestingly, very reduced. This gap can be explained using `JPEGsnoop`, which shows that compression level 100 returns the best-looking samples because **it basically does not perform a step of the compression process**: the quantizing factors used by the luminance and chrominance quantization tables are, in fact, a collection of 1s.

```
DQT, Row #0:   1   1   1   1   1   1   1   1
DQT, Row #1:   1   1   1   1   1   1   1   1
DQT, Row #2:   1   1   1   1   1   1   1   1
DQT, Row #3:   1   1   1   1   1   1   1   1
DQT, Row #4:   1   1   1   1   1   1   1   1
DQT, Row #5:   1   1   1   1   1   1   1   1
DQT, Row #6:   1   1   1   1   1   1   1   1
DQT, Row #7:   1   1   1   1   1   1   1   1

DQT, Row #0:   1   1   1   1   1   1   1   1
DQT, Row #1:   1   1   1   1   1   1   1   1
DQT, Row #2:   1   1   1   1   1   1   1   1
DQT, Row #3:   1   1   1   1   1   1   1   1
DQT, Row #4:   1   1   1   1   1   1   1   1
DQT, Row #5:   1   1   1   1   1   1   1   1
DQT, Row #6:   1   1   1   1   1   1   1   1
DQT, Row #7:   1   1   1   1   1   1   1   1
```

The DCT coefficients are all divided by 1, which means they are **not divided at all**: with this compression quality every small or unnoticeable detail is maintained in the final picture. As for compression level 90, the luma-chroma quantization tables are not even close to the ones used by compression quality 100 as we can see below:

```
DQT, Row #0:   3   2   2   3   5   8  10  12
DQT, Row #1:   2   2   3   4   5  12  12  11
DQT, Row #2:   3   3   3   5   8  11  14  11
DQT, Row #3:   3   3   4   6  10  17  16  12
DQT, Row #4:   4   4   7  11  14  22  21  15
DQT, Row #5:   5   7  11  13  16  21  23  18
DQT, Row #6:  10  13  16  17  21  24  24  20
DQT, Row #7:  14  18  19  20  22  20  21  20

DQT, Row #0:   3   4   5   9  20  20  20  20
DQT, Row #1:   4   4   5  13  20  20  20  20
DQT, Row #2:   5   5  11  20  20  20  20  20
DQT, Row #3:   9  13  20  20  20  20  20  20
DQT, Row #4:  20  20  20  20  20  20  20  20
DQT, Row #5:  20  20  20  20  20  20  20  20
DQT, Row #6:  20  20  20  20  20  20  20  20
DQT, Row #7:  20  20  20  20  20  20  20  20
```

Interestingly, GIMP 2 uses the same standard Huffman tables (12 code words for for the DC component and 162 code words for the AC components) no matter the selected compression quality: since the chroma subsampling setting is chosen by the user then the **difference in size between compression qualities 100 and 90 depends only on their quantization tables**. As for the other compression qualities, we have seen that their results vary depending on the type of picture we are analyzing: for example, sometimes compression level 80 returns very good results (pictorial and textual samples), other times the samples are acceptable but not exactly without defects (gradient samples). Overall, the quality of the compressed images plummets in most cases when the user selects a compression quality lower than 80. To understand why we only need to take a look at the

quantization tables of the samples we have produced. `JPEGsnoop` shows that the **quantizing factors keep growing and growing the more we lower the selected compression quality**[11]. For example, compression quality 70 has the following quantization tables...

```
DQT, Row #0:  10   7   6  10  14  24  31  37
DQT, Row #1:   7   7   8  11  16  35  36  33
DQT, Row #2:   8   8  10  14  24  34  41  34
DQT, Row #3:   8  10  13  17  31  52  48  37
DQT, Row #4:  11  13  22  34  41  65  62  46
DQT, Row #5:  14  21  33  38  49  62  68  55
DQT, Row #6:  29  38  47  52  62  73  72  61
DQT, Row #7:  43  55  57  59  67  60  62  59

DQT, Row #0:  10  11  14  28  59  59  59  59
DQT, Row #1:  11  13  16  40  59  59  59  59
DQT, Row #2:  14  16  34  59  59  59  59  59
DQT, Row #3:  28  40  59  59  59  59  59  59
DQT, Row #4:  59  59  59  59  59  59  59  59
DQT, Row #5:  59  59  59  59  59  59  59  59
DQT, Row #6:  59  59  59  59  59  59  59  59
DQT, Row #7:  59  59  59  59  59  59  59  59
```

...while compression quality 50 has instead:

```
DQT, Row #0:  16  11  10  16  24  40  51  61
DQT, Row #1:  12  12  14  19  26  58  60  55
DQT, Row #2:  14  13  16  24  40  57  69  56
DQT, Row #3:  14  17  22  29  51  87  80  62
DQT, Row #4:  18  22  37  56  68 109 103  77
DQT, Row #5:  24  35  55  64  81 104 113  92
DQT, Row #6:  49  64  78  87 103 121 120 101
DQT, Row #7:  72  92  95  98 112 100 103  99
```

[11] See appendix 4 for the whole transcription of GIMP quantization and Huffman tables.

```
DQT, Row #0:  17  18  24  47  99  99  99  99
DQT, Row #1:  18  21  26  66  99  99  99  99
DQT, Row #2:  24  26  56  99  99  99  99  99
DQT, Row #3:  47  66  99  99  99  99  99  99
DQT, Row #4:  99  99  99  99  99  99  99  99
DQT, Row #5:  99  99  99  99  99  99  99  99
DQT, Row #6:  99  99  99  99  99  99  99  99
DQT, Row #7:  99  99  99  99  99  99  99  99
```

Since every compression quality has the same Huffman tables, the variability in quality and file size shown by our samples depends only on the quantizing factors and on the chroma subsampling settings chosen by the user. To conclude, GIMP's **compression quality numbers are not realistic**: for example, with compression quality 50 the overall quality of the initial image is more than cut in half. To get the best result it is mandatory to **never get below compression quality 85**: while every other compression qualities drastically reduce the file size of the samples, **the resulting quality is not worth it in most cases**.

4.6 Additional case study: different DCT coefficients

As we have said before, GIMP 2 offers three different DCT methods: *float*, *integer* (the default option) and *fast integer*. GIMP 2 online manual describes these methods as follows:

- *Float: the float method is very slightly more accurate than the integer method, but is much slower unless your machine has very fast floating-point hardware. Also note that the results of the floating-point method may vary slightly across machines, while the integer methods should give the same results everywhere.*
- *Integer (the default): this method is faster than float, but not as accurate.*
- *Fast integer: the fast integer method is much less accurate than the other two* [GOM 2015].

To analyse the differences between each method we consider the original high-definition image LEP and compress it with chroma subsampling 4:4:4 to minimize

the potential artefacts. Next, we use an image editor of our choice to analyse each sample. Visually, **samples obtained with the same compression quality appears to be identical** for all intents and purposes: block artefacts, discolourations and posterized areas are exactly the same, especially when lower compression rates are selected (compression qualities between 100 and 50).

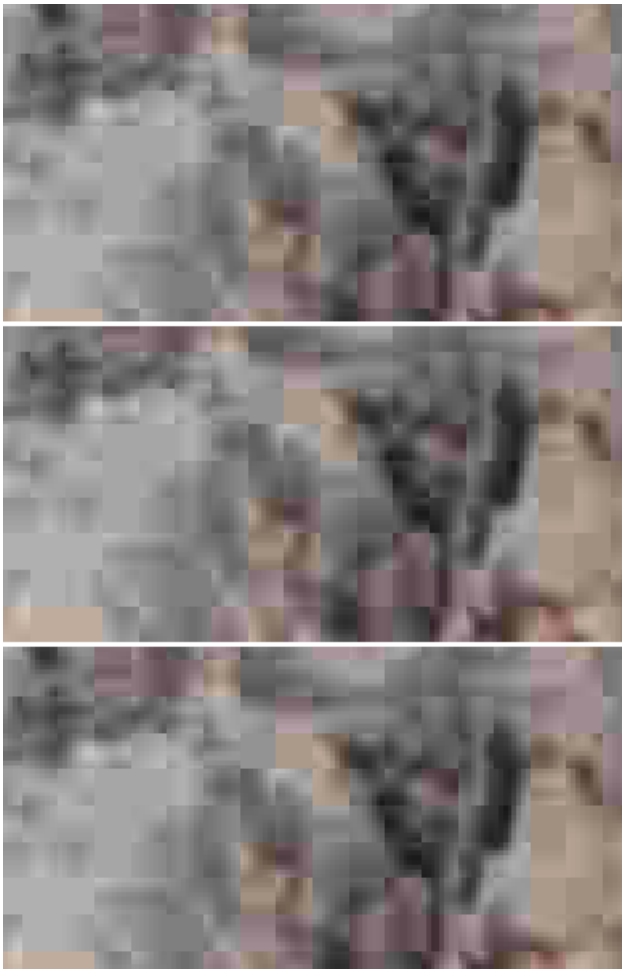

Figure 55: LEP compressed at quality 10 (4:4:4) with integer (top), fast integer and float (bottom) DCT method

Since the quantization tables, Huffman tables and chroma subsampling settings are always the same, we can conclude that different DCT methods mainly impact the *efficiency* of the JPEG compression exactly as stated by the online manual, with no visual consequences on the various samples. As for the **file sizes** of the

samples, we obtain the following results:

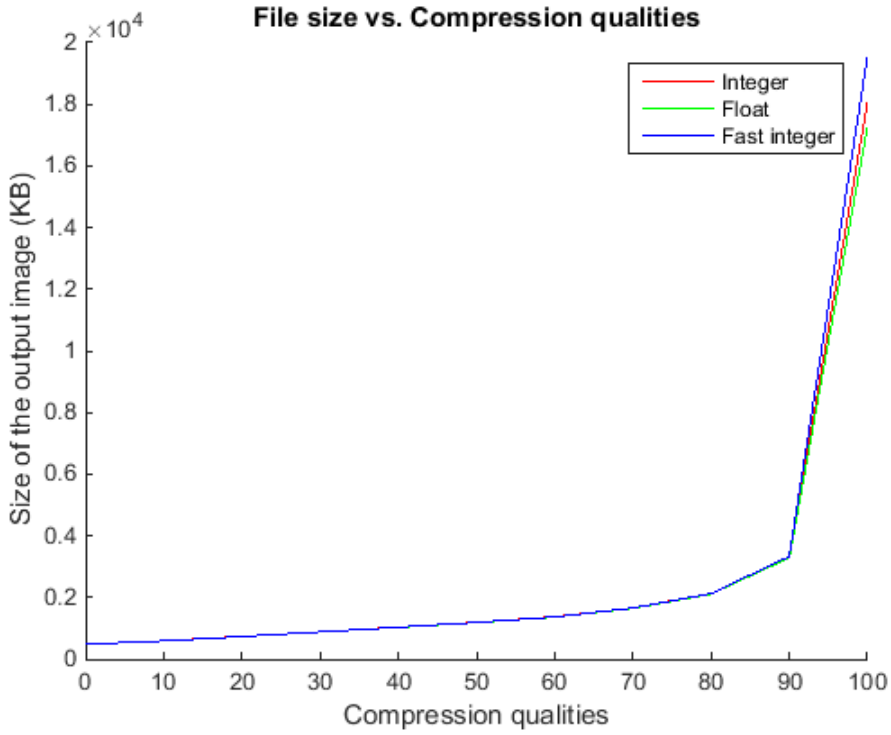

Figure 56: Size vs. compression qualities with different DCT methods.

As we can see, compression qualities 0-90 return samples with almost identical file sizes. Compression quality 100 shows instead more diversity:

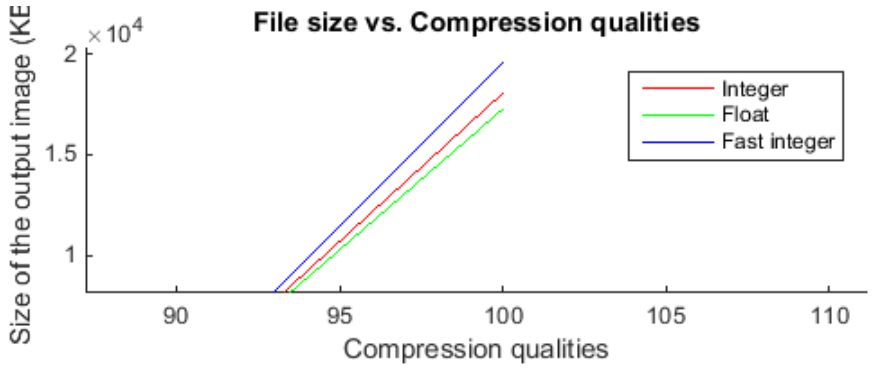

Figure 57: Size vs. compression qualities in details. The data set is available as a stand-alone file called `DifferentDct(Integer,float,fastinteger).txt`

As expected, *fast integer* is the least efficient DCT method: at the highest compression qualities it produces the biggest files, with no visible improvements quality-wise; *float*, instead, performs very well, producing the smallest files. Finally, *integer* stands in the middle, producing average-sized files.

4.7 Additional case study: optimized and progressive compressions

So far we analyzed simple baseline JPEG pictures; however, GIMP 2 offers **two additional options** when exporting a file: optimization and progressive compression. But what does it mean to produce an optimized or progressive JPEG file? To understand what happens when the *progressive* and *optimized* options are ticked we produce 11 progressive and 11 optimized JPEGs using the original high-definition image LEP, with chroma subsampling 4:4:4 and compression qualities varying from 100 to 0. Next, we use an image editor of our choice to analyse each sample. Visually, we notice that there is **not much difference between baseline and optimized or progressive samples**: if we fix the chroma subsampling setting at 4:4:4 and compare samples with the same compression quality we recognize that they look basically the same. In particular, compression qualities between 100 and 80 produce the best-looking samples, with detailed areas, smooth colours and no artefacts. Compression qualities between 70 and 30 still produce good results with mostly sharp edges and realistic colours but show more and more artefacts the more the image is compressed. At compression quality 10 and 0 the samples are instead completely posterized (fig. 56) and thus practically useless. Interestingly, optimized JPEGs are displayed immediately by any image visualizer; it takes instead a couple of seconds to display progressive JPEGs and a low-resolution image is shown at first. Using `JPEGsnoop` we can also see that *progressive* and *optimized* JPEGs share the same quantization tables with their baseline counterparts; they list, instead, different sets of Huffman tables: while progressive JPEGs have different sets Huffman tables for each scan of the picture, optimized JPEGs have one set of custom Huffman tables. In both cases these tables contain **only the code words that are really needed to encode the image (or the scan)** we are working on.

Figure 58: Progressive (left) and baseline JPEGs at compressions 100 (top), 50, 0

As for the **file size**, surprisingly **progressive samples are smaller** than both baseline and optimized JPEGs when compression quality is set at 100 or between 0 and 70; otherwise the samples are more or less on the same level as far as the file size is concerned.

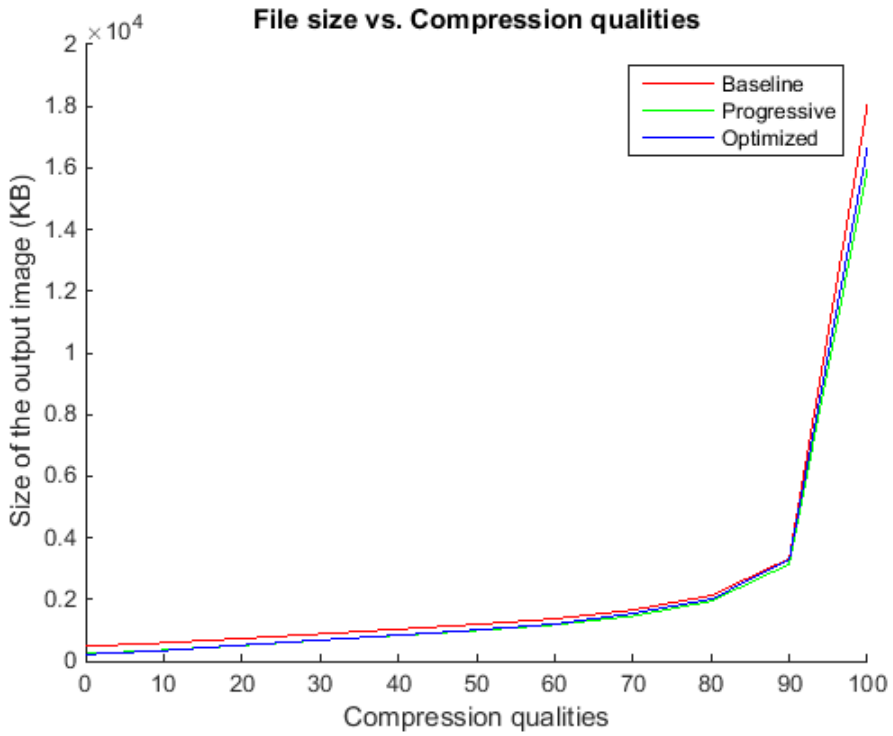

Figure 59: Size vs. compression quality (baseline, progressive and optimized). The data set is available as a stand-alone file called `Progressive.txt`

4.8 GIMP versus Photoshop: a comparison

We have seen in the previous sections that GIMP 2 produces JPEG files with a very wide range of different features depending on the chosen compression quality; the same does not happen when we use Photoshop CS5: while its compressed files do show discolourations, colour bands and block artefacts, they are never as posterized and ruined as those produced by GIMP 2, even at the lowest possible compression levels. Therefore it appears that the comparison between Photoshop CS5 and GIMP 2 JPEG compressions **can be made only if we consider GIMP's best compression qualities**, between 100 and 60: every other setting is too low-quality to be used in this comparison.

4.8.1 `Save as` versus `Export`: visual comparison

The first criterion we will use to match Photoshop and GIMP JPEG files is **visual similarity**: if two samples have the same amount of block artefacts, discolourations, posterized areas or colour bands then a match is found. We obtain the following results:

- **Best quality**. Photoshop compression level 12 and GIMP compression qualities 100 (4:4:4), 99 (4:4:4) and 98 (4:4:4) seem to return the same high-quality JPEG files, with very sharp edges, smooth colour transitions and no block artefacts;

- **Better quality.** Photoshop compression level 11 can be matched to GIMP compression qualities 97 (4:4:4) and 96 (4:4:4): the overall quality of the samples is still very high but if we zoom enough we can see some very minor artefacts;

- **Average quality**. Photoshop compression level 10 and GIMP compression qualities 95 (4:4:4), 94 (4:4:4) and 93 (4:4:4) appear to be closely related: the artefacts are way more prominent in every sample we analyse, but are still unnoticeable if we do not zoom on each picture. Photoshop compression level 9 can be matched with GIMP compression quality 92 (4:4:4) since the amount of discolourations appears to be almost the same. Photoshop compression level 8 is matched with GIMP compression quality 91 (4:4:4): artefacts are more and more visible in both samples, especially on edges and smooth surfaces. Finally, Photoshop compression level 7 and GIMP compression quality 90 (4:4:4) are extremely similar since they show the same amount of visible block artefacts.

- **Lower quality**. Photoshop compression level 6 and GIMP compression quality 91 (4:2:2 horizontal) are closely related: block artefacts are visible in both samples but the overall quality of the pictures is still acceptable. Note that Photoshop compression level 6 is matched with GIMP compression quality 91 — higher than the compression quality used for Photoshop compression level 7: this is completely in line with the observations made on the peculiar gap between Photoshop compression levels 7 and 6 (see section

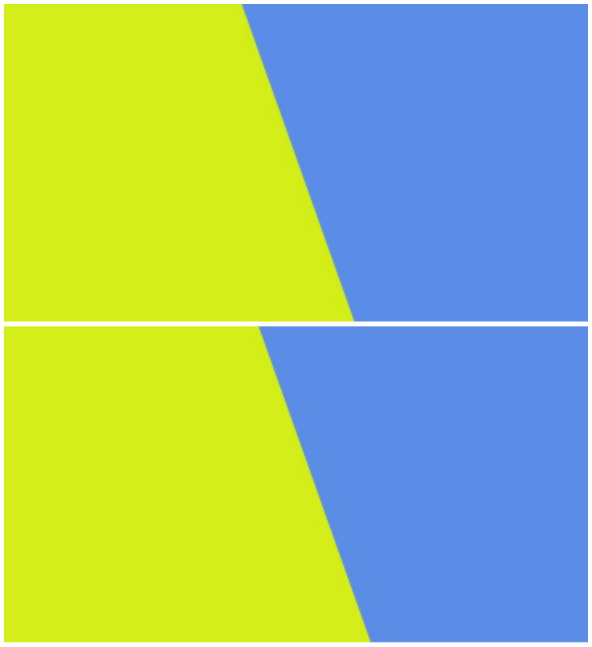

Figure 60: `GNS2C` compressed by Photoshop (top) and GIMP (bottom), at compression levels 12 and 98 respectively.

3.5.2). Photoshop compression level 5 can be matched to GIMP compression qualities 90 (4:2:2 horizontal) and 89 (4:2:2 horizontal): the corresponding samples show artefacts of comparable size and features. Photoshop compression level 4 and GIMP compression qualities 88 (4:2:2 horizontal), 87 (4:2:2 horizontal) and 86 (4:2:2 horizontal) are very similar: block artefacts, discolourations and colour bands are more and more visible; Finally, Photoshop compression level 3 can be matched with GIMP compression qualities 85 (4:2:2 horizontal) and 84 (4:2:2 horizontal).

- **Worst quality**. Photoshop compression level 2 can be matched with GIMP compression quality 83 (4:2:2 horizontal); Photoshop compression level 1 can be matched instead with GIMP compression qualities 82 (4:2:2 horizontal), 81 (4:2:2 horizontal) and 80 (4:2:2 horizontal). Finally, Photoshop compression level 0 can be matched with GIMP compression qualities 70 (4:2:2 horizontal) and lower: in both cases blocks are *very* visible, with many discolourations, posterized areas and colour bands.

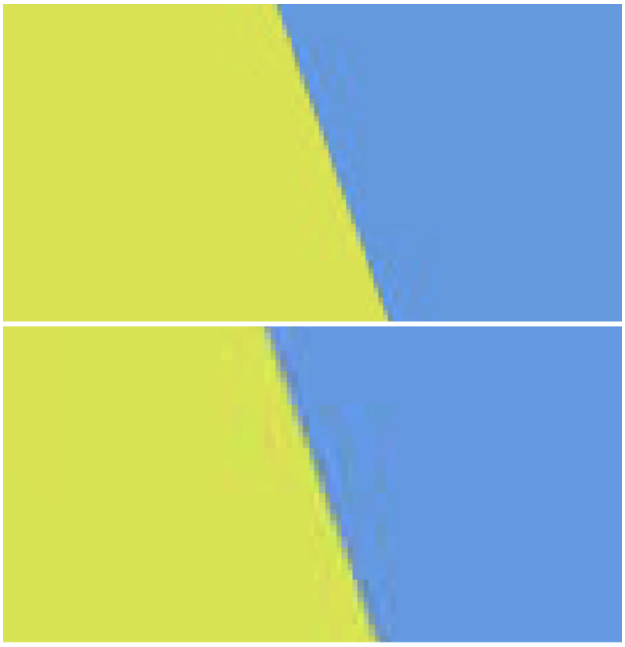

Figure 61: `GNS2C` compressed by Photoshop (top) and GIMP (bottom), at compression levels 5 and 90 (4:2:2 horizontal) respectively.

Photoshop levels	Gimp qualities
12	100, 99, 98 (4:4:4)
11	97, 96 (4:4:4)
10	95, 94, 93 (4:4:4)
9	92 (4:4:4)
8	91 (4:4:4)
7	90 (4:4:4)
6	91 (4:2:2 h)
5	90, 89 (4:2:2 h)
4	88, 87 (4:2:2 h)
3	85, 84 (4:2:2 h)
2	83 (4:2:2 h)
1	82, 81 (4:2:2 h)
0	70 (4:2:2 h)

As we can see, this comparison shows that the gap between Photoshop compression levels 1 and 0 is and that most of the compression qualities offered by GIMP must be discarded: compression qualities under 70 are in fact too low-quality to be compared with Photoshop compression levels.

4.8.2 `Save as` vs. `Export`: data stream

To assess the quality of our previous comparison we can use `JPEGsnoop` and compare quantization tables, Huffman tables and chroma subsampling settings used by each compression level. We have the following results[12]:

- **Best quality**. GIMP compression quality 100 (4:4:4) is way better than every other setting offered by Photoshop `Save as` since it uses **quantization tables with the lowest possible coefficients**. Since both compressions use the same chroma subsampling settings and Huffman tables, it is incorrect to say that GIMP compression quality 100 approximates Photoshop compression level 12: `JPEGsnoop` shows that it is actually the other way around. GIMP compression quality 99 (4:4:4) is still better than Photoshop compression level 12, since its luma-chroma quantization tables have lower coefficients. Again, since both compressions use the same chroma subsampling settings and Huffman tables, GIMP seems to be the best choice. GIMP compression quality 98 (4:4:4) is instead very similar but ultimately worse than Photoshop compression level 12, since its luma-chroma quantization tables have higher coefficients; still, it is better than Photoshop compression level 11, whose coefficients are way smaller than those used by GIMP.

- **Better quality**. GIMP compression level 97 (4:4:4) is actually better than Photoshop compression level 11 since its luma-chroma quantization tables have smaller coefficients; GIMP compression quality 96 (4:4:4) is instead worse than Photoshop compression level 11 but better than Photoshop compression level 10.

- **Average quality**. GIMP compression qualities 95, 94 and 93 (4:4:4) are comparable to Photoshop compression level 10 (GIMP 95 is slightly better,

[12]See appendices 1 and 4 to get the complete listing of quantization tables, Huffman tables and chroma subsampling settings.

GIMP 94 is on the same level and 93 is slightly worse than Photoshop 10); GIMP compression qualities 92 and 91 (4:4:4) are generally worse than Photoshop compression level 9 and 8 respectively; GIMP compression quality 90 (4:4:4) is instead *way* worse than Photoshop compression level 7, despite being similar visually.

- **Lower quality**. As we already know, Photoshop compression level 6 is a bit of a "wild card", with very good quantization tables and a stricter chroma subsampling setting, 2×2. Visually, it can be compared with GIMP compression quality 91 (4:2:2 horizontal); if we consider only the quantization and Huffman tables the same compression quality can be associated with Photoshop compression level 8 or 7. Still, the quantization tables used in all these compressions are very different and a precise match is difficult to find. While Photoshop compression level 5 can be compared to compression qualities 90 and 89 (4:2:2 horizontal) from a visual point of view, the quantization tables of the latter compressions have very high coefficients — higher than those used by Photoshop compression level 5. The same can be said for Photoshop compression level 4 and GIMP compression qualities 88 and 87 (4:2:2 horizontal): despite being visually similar, their quantization tables are fairly different. Finally, Photoshop compression level 3 is more or less on the same level as GIMP compression qualities 85 and 84 (4:2:2 horizontal), since their quantization tables are very similar.

- **Worst quality**. Photoshop compression level 2 is approximated by GIMP compression quality 83 (4:2:2 horizontal) while Photoshop compression level 1 is similar to GIMP compression qualities 82 and 81 (4:2:2 horizontal). As for Photoshop compression level 0, visually it can be approximated by GIMP compression quality 70 (4:2:2 horizontal) but if we consider the corresponding quantization tables we realize that GIMP compression quality 72 (4:4:2 horizontal) is a better match. Still, it is difficult to compare these quantization tables as Photoshop `Save as` and GIMP `Export` seem to use two different compression strategies: while Photoshop's quantization tables tend to have higher DC components and smaller AC components, GIMP uses the inverse methodology, with smaller DC components and very high AC components.

As we can see, our **previous visual comparison has to be slightly adjusted**; still, it was mostly accurate. To sum up, to get the overall best results it is wise to choose GIMP compression qualities 100, 99 or 98 with chroma subsampling 4:4:4; Photoshop compression levels 11-6 are visually better than their GIMP counterparts; however, their file sizes are usually bigger than than those of the files produced by GIMP. As for the lower compression levels, GIMP and Photoshop return very similar results and no application is better than the other; still, it is important to remember that most compression qualities offered by GIMP 2 are not even considered in this comparison: their samples are too low-quality to be used. Finally, notice that the default compression quality suggested by GIMP is 85, which corresponds to one of the lowest levels offered by Photoshop `Save as` feature, 3. As for the **file sizes**, suppose we consider GNS2C; the results are the following:

Photoshop file size	Gimp file size
12, 1041KB	98 (4:4:4), 541KB
11, 835KB	96 (4:4:4), 502KB
10, 495KB	93 (4:4:4), 466KB
9, 473KB	92 (4:4:4), 458KB
8, 455KB	91 (4:4:4), 454KB
7, 439KB	90 (4:4:4), 448KB
6, 285KB	91 (4:2:2 h), 343KB
5, 263KB	90 (4:2:2 h), 338KB
4, 256KB	89 (4:2:2 h), 333KB
3, 252KB	88 (4:2:2 h), 330KB
2, 181KB	87 (4:2:2 h), 326KB
1, 175KB	86 (4:2:2 h), 323KB
0, 173KB	85 (4:2:2 h), 320KB

As we can see, sizes are very different: while Photoshop produces the biggest files when high compression levels are selected, this trend is reversed when the lowest compression levels are chosen instead.

4.8.3 `Save for web` vs. `Export`: visual comparison

Photoshop `Save for web` method offers 101 different compression qualities, exactly as GIMP 2; still, the results are widely different both visually and technically. In fact, the visual comparison between samples produced by Photoshop CS5 `Save for Web` and GIMP 2 yields the following results:

- **Maximum quality**. Photoshop compression quality 100 and GIMP compression qualities 100, 99 and 98 (4:4:4) seem to return the same high-quality samples: edges are very sharp, artefacts are neither present nor visible, colours are smooth with no bands or posterized areas.
- **Very high quality**. Photoshop compression qualities 99-80 can be matched to GIMP compression qualities 97-93 (4:4:4): block artefacts are present but almost unnoticeable, edges still look sharp and colours do not show bands or peculiar hues.

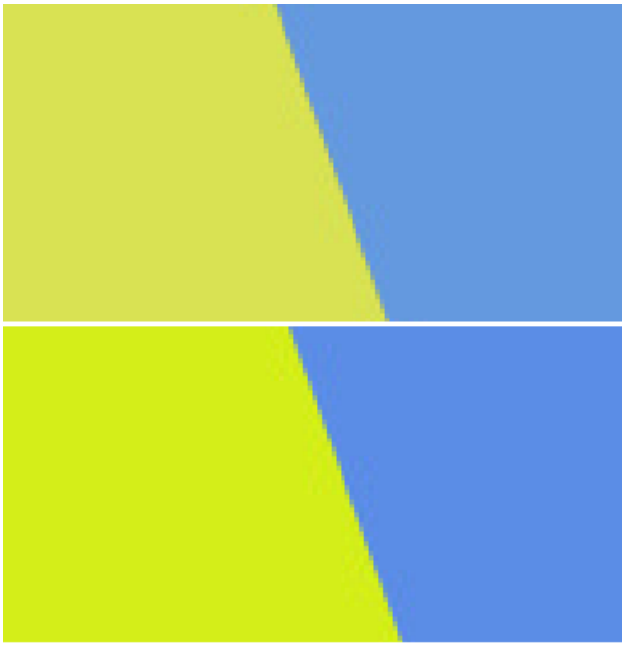

Figure 62: `GNS2C` compressed by Photoshop (top) and GIMP (bottom), at compression levels 80 and 93 (4:4:4).

- **High quality**. Photoshop compression qualities 79-60 and GIMP compression qualities 92-85 (4:4:4) yields similar results: block artefacts are more

and more visible when zooming, but still unnoticeable when looking at the thumbnails of the samples.

- **Medium quality**. Photoshop compression qualities 59-51 can be roughly matched to GIMP compression qualities 84-80 (4:4:4) while Photoshop compression qualities 50-30 can be compared to GIMP compression qualities 86-74 (4:2:2 horizontal). As we can see, it is complex to find a perfect match for this set of compression qualities since there is a change of chroma subsampling settings, from 4:4:4 to 4:2:2.

- **Low quality**. Photoshop compression qualities 29-0 and GIMP compressions 73-38 (4:2:2 horizontal) seem to share the same amount of block artefacts, colour bands and posterized areas, and thus can be linked together.

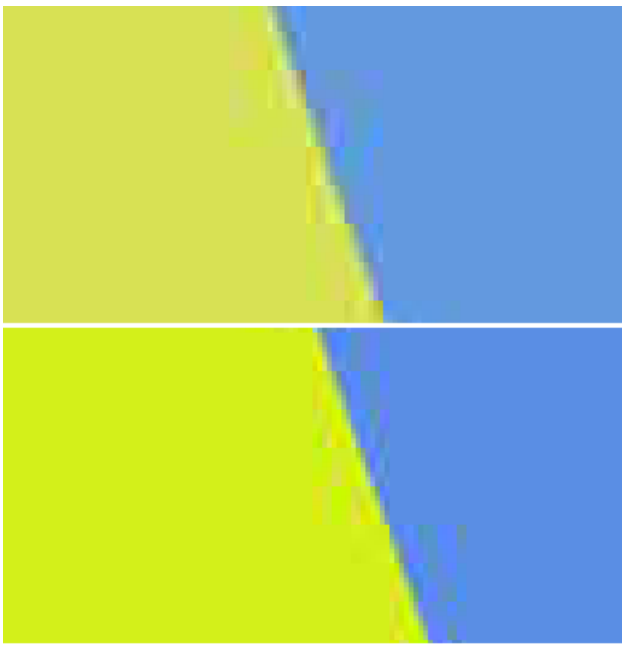

Figure 63: `GNS2C` compressed by Photoshop (top) and GIMP (bottom), at compression levels 0 and 38 (4:2:2 horizontal).

As before, this comparison shows there is a consistent gap between Photoshop and GIMP compression qualities, with **Photoshop delivering comparably better samples even at its lowest compressions levels**. The same does not apply

Photoshop level	Gimp level
100	100-98 (4:4:4)
99-80	97-93 (4:4:4)
79-60	92-85 (4:4:4)
59-30	84-80 (4:4:4), 86-74 (4:2:2 h)
29-0	73-38 (4:2:2 h)

to GIMP, since its lowest compression qualities are too low-quality to be even considered in this comparison.

4.8.4 `Save for web` vs. Export: data stream

As before, to evaluate the quality of our previous comparison we can use `JPEGsnoop` and compare quantization tables, Huffman tables and chroma subsampling settings. The results are the following[13]

- **Best quality**. GIMP compression quality 100 (4:4:4) is still better than Photoshop compression quality 100: GIMP uses better quantization tables with the lowest possible coefficients. Photoshop `Save for web` compression quality 100 can more or less be matched to GIMP compression qualities 99, and it is definitely better than GIMP compression quality 98 (4:4:4), which is in fact more similar to GIMP compression level 99 (4:4:4).

- **Very high quality**. Photoshop compression qualities 99-80 and GIMP compression qualities 99-94 (4:4:4) share very similar quantization tables and can thus be matched. We previously stated that Photoshop compression quality 80 was very similar to GIMP compression quality 93, but apparently we were mistaken.

- **High quality**. Photoshop compression qualities 79-60 can be matched to GIMP compression qualities 93-85; in particular, GIMP compression quality 85 is only slightly better than Photoshop compression quality 60.

[13] Cfr. appendices 2 and 4 to get the full listing of quantization tables, Huffman tables and chroma subsampling settings used in these comparisons.

- **Medium quality**. We have said before that, visually, Photoshop compression qualities 59-51 can be matched to GIMP compression qualities 84-80 (4:4:4) while Photoshop compression qualities 50-30 can be compared with GIMP compression qualities 86-74 (4:2:2 horizontal). These comparisons seems to be accurate when we consider the quantization tables, chroma subsampling settings and Huffman tables of these compressions.

- **Low quality**. Photoshop compression qualities 29-0 can be compared to GIMP compression qualities 73-60 (4:2:2 horizontal): as we can see Photoshop compression quality 0 is here matched with a relatively high GIMP compression quality, 60 (4:2:2 horizontal), if we are concerned only with quantization tables, Huffman tables and chroma subsampling settings.

Our previous comparisons were mostly accurate, with **the only exception being Photoshop compression level 0**: visually it can be matched with GIMP compression quality 38 but, if we look only at quantization tables, Huffman tables and chroma subsampling settings, GIMP compression quality 60 (4:2:2) seems to be a better choice.

Photoshop level	Gimp level
100	99, 98 (4:4:4)
99-80	97-94 (4:4:4)
79-60	93-85 (4:4:4)
59-51, 50-30	84-80 (4:4:4), 86-74 (4:2:2 h)
29-0	73-60 (4:2:2 h)

Interestingly, when comparing GIMP and Photoshop `Save for web`, we use an extended set of GIMP compression qualities, going as far as comparing Photoshop compression quality 0 to GIMP compression quality 38 and 60; still, many compression qualities offered by GIMP are again discarded since there is no match between GIMP's and Photoshop's lower compression qualities. Note also that the previous table underlines how GIMP focuses on average-to-low quality samples: while Photoshop has more or less 30 levels to render low-quality images, GIMP has instead more than 70 quality levels. Finally, when the **file size** is concerned, if

we consider again GNS2C we notice that there is no real correspondence between Photoshop CS5 `Save for Web` and GIMP 2:

Photoshop level	Gimp level
100 (1020KB)	99-98 (4:4:4) (618-582KB)
99-80 (1017-480KB)	97-94 (4:4:4) (520-476KB)
79-60 (477-433KB)	93-85 (4:4:4) (466-320KB)
59-51 (432-425KB),	84-80 (4:4:4) (423-415KB),
50-30 (272-241KB)	86-74 (4:2:2 h) (323-300KB)
29-0 (241-151KB)	73-60 (4:2:2 h) (298-288KB)

As we can see, at the highest compression qualities GIMP produces the smallest files; the trend is reversed when we consider the lowest compression qualities.

To sum up, GIMP 2 JPEG compression always reduces the size of the original high-definition images, sometimes drastically so when the highest compression rates are selected. Compression quality 100 performs extremely well in every possible situation, as noted before; the other compression qualities are instead very good at dealing with pictorial images but show many artefacts when they try to compress geometrical, textual and gradient samples, especially when the selected compression quality is very low.

5 JPEG on the web

JPEG is by far one of the most used image file format on the web: with its selectable trade-off between quality and file size its rise comes as no surprise, especially if we consider how many websites are now offering backup services for digital media or online galleries. Still, sometimes the files we see on websites such as Facebook or Flickr are **not exactly identical to the ones we uploaded**: further compressions are usually at work and the results are often underwhelming to say the least. In the following pages we will analyse the JPEG files produced by three different websites — Facebook, Google Photos and Flickr — with the goal of understanding what happens when a user uploads an image and which processes are undergone by our pictures.

5.1 Facebook

Facebook is an online social networking service founded in 2004 and nowadays counting more than 1 billion of monthly active users. Registered users can create a personal profile page, update their status, add other users as friends, exchange messages, share digital contents (photos or videos), join groups, follow pages or power users, etc. One of the perks offered by Facebook is **free image storage space** — there is no limit to the number of images any user can upload to their personal profile.

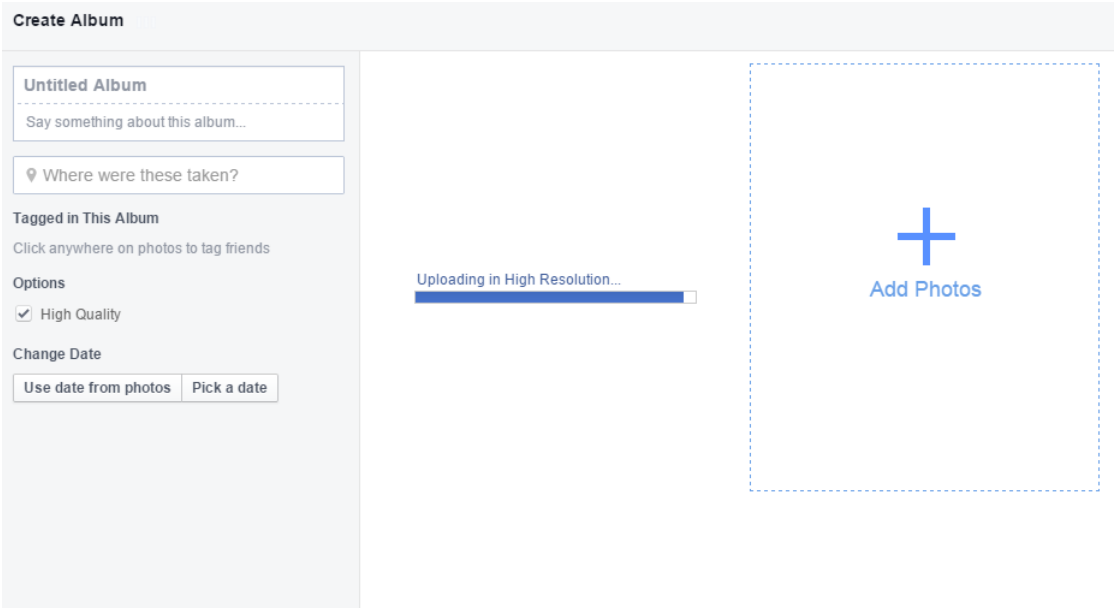

Figure 64: Facebook upload panel

This service, however, has a severe limitation: **every picture is inevitably re-compressed**. This means that an otherwise crystalline picture may or may not show artefacts when uploaded on Facebook: while this can be a minor nuisance for most users, it can become a problem for commercial brands trying to showcase their businesses. But what kind of compression is performed during the upload? Facebook offers two different options: `standard` and `high quality` upload (fig. 64); to understand how they are structured we will analyse two different pictures, GNS2C and LLP produced using Photoshop CS5 `Save as` compression level 12:

Figure 65: LLP and GNS2C used to test Facebook compression

As noted before this compression uses standard Huffman tables, no chroma subsampling and two very good luma-chroma quantization tables, which are listed below:

```
DQT, Row #0:   1   1   1   1   1   1   1   2
DQT, Row #1:   1   1   1   1   1   1   1   2
DQT, Row #2:   1   1   1   1   1   1   2   2
DQT, Row #3:   1   1   1   1   1   2   2   3
DQT, Row #4:   1   1   1   1   2   2   3   3
DQT, Row #5:   1   1   1   2   2   3   3   3
DQT, Row #6:   1   1   2   2   3   3   3   3
DQT, Row #7:   2   2   2   3   3   3   3   3

DQT, Row #0:   1   1   1   2   3   3   3   3
DQT, Row #1:   1   1   1   2   3   3   3   3
DQT, Row #2:   1   1   2   3   3   3   3   3
DQT, Row #3:   2   2   3   3   3   3   3   3
DQT, Row #4:   3   3   3   3   3   3   3   3
DQT, Row #5:   3   3   3   3   3   3   3   3
DQT, Row #6:   3   3   3   3   3   3   3   3
DQT, Row #7:   3   3   3   3   3   3   3   3
```

5.1.1 Standard upload

To see how the samples are affected by Facebook standard compression we upload them on Facebook and re-download the pictures; next, we use `JPEGsnoop`

to evaluate them. The first thing we notice is that Facebook re-compresses the samples discarding any additional marker used by Photoshop, to reduce the file size. The quantization tables for the luminance and chrominance components are also completely different:

```
DQT, Row #0:   9   6   6   9  14  23  30  35
DQT, Row #1:   7   7   8  11  15  34  35  32
DQT, Row #2:   8   8   9  14  23  33  40  32
DQT, Row #3:   8  10  13  17  30  50  46  36
DQT, Row #4:  10  13  21  32  39  63  60  45
DQT, Row #5:  14  20  32  37  47  60  66  53
DQT, Row #6:  28  37  45  50  60  70  70  59
DQT, Row #7:  42  53  55  57  65  58  60  57

DQT, Row #0:  10  10  14  27  57  57  57  57
DQT, Row #1:  10  12  15  38  57  57  57  57
DQT, Row #2:  14  15  32  57  57  57  57  57
DQT, Row #3:  27  38  57  57  57  57  57  57
DQT, Row #4:  57  57  57  57  57  57  57  57
DQT, Row #5:  57  57  57  57  57  57  57  57
DQT, Row #6:  57  57  57  57  57  57  57  57
DQT, Row #7:  57  57  57  57  57  57  57  57
```

As we can see the coefficients are very high: in fact they can be **compared to those of Photoshop** `Save for web` **compression quality 40**. Chroma subsampling is also performed:

```
Component[1]: ... (Subsamp 1 x 1), ... (Lum: Y)
Component[2]: ... (Subsamp 2 x 2), ... (Chrom: Cb)
Component[3]: ... (Subsamp 2 x 2), ... (Chrom: Cr)
```

Finally, the resulting images are **progressive JPEGs** as `JPEGsnoop` shows different scans, each with a **different set of Huffman tables**. In particular, these Huffman tables contain a *very* limited number of code words. The files' dimensions are also **reduced**:

File	Original size (pixels)	Facebook size (pixels)
LLP	3872 × 2592	960 × 643
GNS2C	4257 × 3000	960 × 677

As for the file size, Facebook performs a drastic reduction:

File	Original size	Facebook size
LLP	4936KB	17KB
GNS2C	1041KB	15KB

It goes without saying that a visual comparison of the original samples versus the files produced by Facebook shows all the limitations of this compression: edges are not sharp, block artefacts are very visible and colours are unnaturally altered.

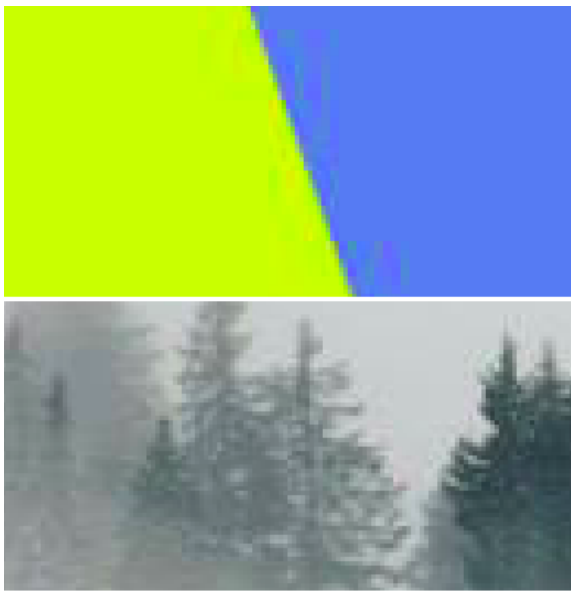

Figure 66: GNS2C and LLP compressed by Facebook, zooming at 200%

5.1.2 High quality upload

Let's repeat the previous experiment and upload both LLP and GNS2C compressed using Photoshop `Save as` setting (compression level 12) on Facebook, ticking the "High quality" option. Next, we download these pictures and use `JPEGsnoop` to assess them. Again, we notice that Facebook discards any additional marker used by Photoshop, again to reduce the file size. The quantization

tables for the luminance and chrominance components are identical to the ones used for the standard compression:

```
DQT, Row #0:   9   6   6   9  14  23  30  35
DQT, Row #1:   7   7   8  11  15  34  35  32
DQT, Row #2:   8   8   9  14  23  33  40  32
DQT, Row #3:   8  10  13  17  30  50  46  36
DQT, Row #4:  10  13  21  32  39  63  60  45
DQT, Row #5:  14  20  32  37  47  60  66  53
DQT, Row #6:  28  37  45  50  60  70  70  59
DQT, Row #7:  42  53  55  57  65  58  60  57

DQT, Row #0:  10  10  14  27  57  57  57  57
DQT, Row #1:  10  12  15  38  57  57  57  57
DQT, Row #2:  14  15  32  57  57  57  57  57
DQT, Row #3:  27  38  57  57  57  57  57  57
DQT, Row #4:  57  57  57  57  57  57  57  57
DQT, Row #5:  57  57  57  57  57  57  57  57
DQT, Row #6:  57  57  57  57  57  57  57  57
DQT, Row #7:  57  57  57  57  57  57  57  57
```

Again, these coefficients are very high and can be compared to those used by Photoshop `Save for web` compression quality 40. Chroma subsampling is performed as before:

```
Component[1]: ... (Subsamp 1 x 1), ... (Lum: Y)
Component[2]: ... (Subsamp 2 x 2), ... (Chrom: Cb)
Component[3]: ... (Subsamp 2 x 2), ... (Chrom: Cr)
```

The resulting images are again progressive JPEGs: `JPEGsnoop` shows several scans, each with a different set of Huffman tables — which are incidentally identical to the ones used by the standard compression. Furthermore, the files' dimensions are not as reduced as before:

File	Original size (pixels)	Facebook size (pixels)
LLP	3872 × 2592	2048 × 1444
GNS2C	4257 × 3000	2048 × 1371

The file sizes are instead strongly reduced:

File	Original size	Facebook size
LLP	4936KB	131KB
GNS2C	1041KB	42KB

A visual comparison of these high quality samples versus the original high-definition pictures shows that, despite the "high quality" label, the samples are still flawed: artefact, discolourations and bands are still very visible. Still, they are better than the standard Facebook files analysed above: the high quality samples have slightly sharper edges, colours are slightly smoother, blocks are still mostly noticeable.

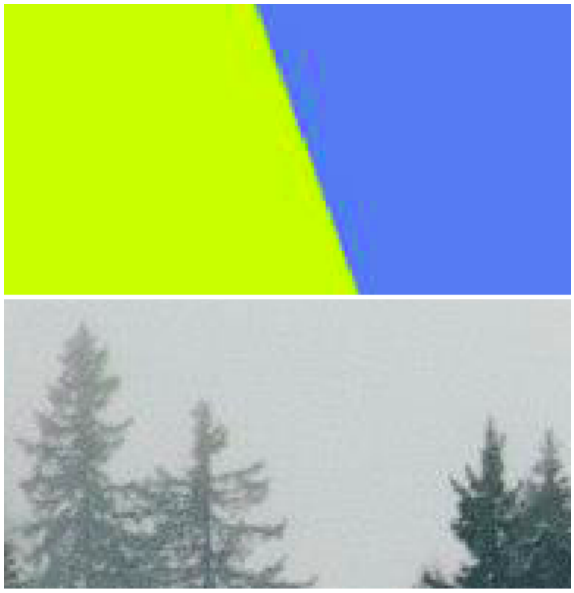

Figure 67: GNS2C and LLP compressed by Facebook (High quality), zooming at 200%

The high quality samples look better, though, because of their increased size: since Facebook displays digital contents using a small black panel (fig. 68), samples appear to be of a higher quality when in fact the difference between these settings is minimal. To sum up, there is **not much difference between standard and high quality settings**; still, the high quality option seems to be the go-to setting when uploading pictures on Facebook, if only for the increased resolution. However, the high quality option is not offered when we upload images directly into

a status update, cover picture or profile picture. Interestingly, Facebook is aware of the limitations caused by these compressions and has issued its own guidance[14] suggesting the following solutions:

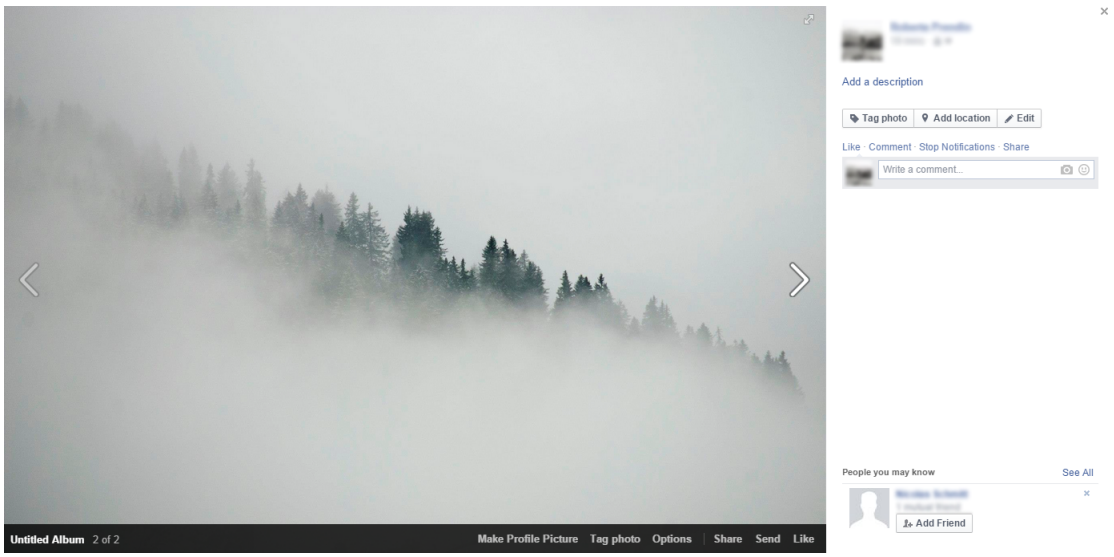

Figure 68: Facebook panel showcasing LLP

- Resize photos to one of the following supported sizes: 720px, 960px, 2048px (width) for regular photos, 851px by 315px for cover photos;
- Select the High Quality option when using a 2048px photo;
- To avoid compression when uploading a cover photo, make sure the file size is less than 100KB;
- Always use an sRGB colour profile.

Still, if we upload the image shown below (size 95KB, fig. 69) compression seems to be performed regardless since the resulting image is a progressive JPEG with the same chroma subsampling settings, quantization tables and Huffman tables seen in the previous pages. All in all, Facebook always performs some sort of

[14]For more information, `https://www.facebook.com/help/266520536764594`.

compression no matter the image we upload; a temporary solution suggested by many users is **switching to another file format**, for example PNG.

Figure 69: The sample used to test Facebook's suggestions

5.2 Google Photos

Google Photos is a digital content (videos and images) sharing and storage service offered by Google. Originally part of the social network Google+, nowadays it is a standalone service that includes unlimited high-quality photo and video storage as well as apps for Android and iOS. Interestingly, the Photos service analyses images and groups them according to their features (mountains, beaches, people, etc). When uploading a photo or a video for the first time the user has to choose between two options: "high quality" and "original". They are described by the *Google Photos help service* as follows:

> **High quality**
>
> - Unlimited free storage
> - Regular cameras: Recommended for phones or point-and-shoot cameras that are 16 megapixels (MP) or less.
> - Uses: Good for typical printing and sharing.
> - Size: Save high-quality photos and videos while reducing size.

Original

- Limited free storage: Uses your Google Account's 15 GB of free storage
- DSLR cameras: Recommended if you take photos with a DSLR camera and want to maintain the exact original quality.
- Uses: Recommended for printing large banners or to store your original files.
- Size: Store your photos and videos exactly as you captured them.

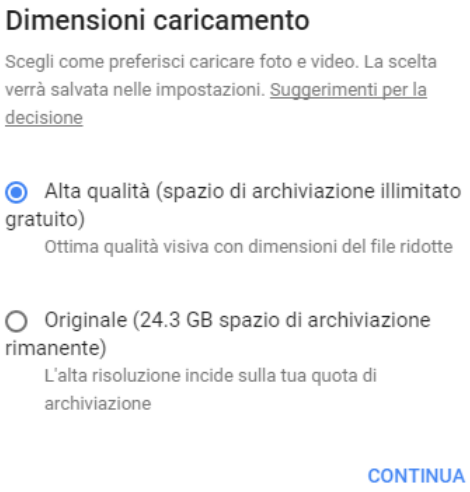

Figure 70: Uploading options in Google Photos (Italian version)

Indeed, the *original* option preserves the original content of the image: the resulting file has the same size and resolution, and shows the same quantization tables, Huffman tables and chroma subsampling settings of the uploaded image. The situation is different for the *high quality* option, which performs some sort of compression to reduce the size of the file. In the following sections we will focus on the latter option and see its results.

5.2.1 High quality

To understand what kind of compression Google Photos performs, we upload both GNS2C and LLP selecting the "high quality" option; as before, both samples are produced using Photoshop CS5 `Save as`, compression level 12. Next, we download both pictures and compare the results using `JPEGsnoop`.
The first thing we notice is that the downloaded files **maintain the additional markers** used by Photoshop CS5: unlike Facebook where files contain only bare information, Google Photos seems to suggest we are indeed using a backup service rather than a social functionality. As for the quantization tables, the luminance one is the following:

```
DQT, Row #0:   3   2   2   3   5   8  10  12
DQT, Row #1:   2   2   3   4   5  12  12  11
DQT, Row #2:   3   3   3   5   8  11  14  11
DQT, Row #3:   3   3   4   6  10  17  16  12
DQT, Row #4:   4   4   7  11  14  22  21  15
DQT, Row #5:   5   7  11  13  16  21  23  18
DQT, Row #6:  10  13  16  17  21  24  24  20
DQT, Row #7:  14  18  19  20  22  20  21  20
```

The quantization tables for chrominance is instead:

```
DQT, Row #0:   3   4   5   9  20  20  20  20
DQT, Row #1:   4   4   5  13  20  20  20  20
DQT, Row #2:   5   5  11  20  20  20  20  20
DQT, Row #3:   9  13  20  20  20  20  20  20
DQT, Row #4:  20  20  20  20  20  20  20  20
DQT, Row #5:  20  20  20  20  20  20  20  20
DQT, Row #6:  20  20  20  20  20  20  20  20
DQT, Row #7:  20  20  20  20  20  20  20  20
```

As we can see Google Photos re-compresses the uploaded images and produces files whose compression can be compared to **Photoshop CS5 `Save for web` compression quality 70 or GIMP 2 compression quality 90**. The coefficients, in fact, are not extremely high: while unnoticeable details are not maintained, the

overall quality of the picture is not overly reduced. Chroma subsampling is also performed:

```
Component[1]: ... (Subsamp 1 x 1), ... (Lum: Y)
Component[2]: ... (Subsamp 2 x 2), ... (Chrom: Cb)
Component[3]: ... (Subsamp 2 x 2), ... (Chrom: Cr)
```

The resulting images are *not* progressive JPEG — an understandable choice since progressive compression does not make sense for a backup cloud service such as Google Photos. Furthermore, Google Photos uses the standard Huffman tables also employed by Photoshop and GIMP baseline JPEGs. As for the resolution of the files, we collect the following data:

File	Original size (pixels)	Facebook size (pixels)
LLP	3872 × 2592	3872 × 2592
GNS2C	4257 × 3000	4257 × 3000

As we can see the samples are not resized; still, the file size is reduced but not as much as with Facebook:

File	Original size	Facebook size
LLP	4936KB	1637KB
GNS2C	1041KB	277KB

A visual comparison (fig. 71) of the original samples versus the files produced by Google Photos shows a very reasonable amount of block artefacts, which are mostly unnoticeable. Critical areas (gradient colours, shadows, small details, ...) do not suffer from the compression: colours are not unnatural, there are no posterized areas and edges appear to be very sharp.

Figure 71: GNS2C and LLP compressed by Google Photos (High quality)

To sum up, the *high quality* option is a very good trade-off between file size and picture quality, and therefore the best solution if we are not looking to back up our files but simply to have a copy of them online.

5.3 Flickr

Flickr is a well-known image and video hosting website created by Ludicorp in 2004 and later acquired by Yahoo in 2005. Nowadays it supports JPEGs, non-animated GIFs, PNGs and TIFFs (that are automatically converted to and stored in JPEG format) and counts millions of pictures uploaded daily. Interestingly, Flickr does not offer many options when uploading pictures, as least when quality is concerned: while it is possible to edit the title, the description, the copyright and the tag associated to the file,the quality of the picture cannot be set by the user.

Figure 72: Flickr "uploadr" panel

Furthermore, once the upload is completed downloading the resulting samples is not exactly as easy as one might expect: since Flickr's target audience is mainly composed by photographers and enthusiasts, downloading pictures is highly discouraged to avoid copyright infringements. Still, it is possible to grab the address of the image by looking at its sharing panel, and to select the size of the sample we want to analyse:

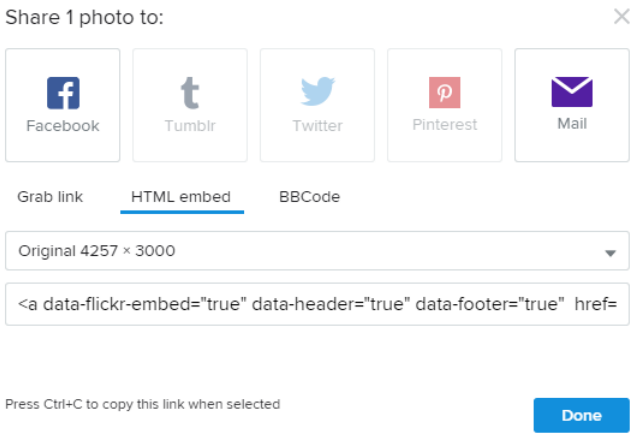

Figure 73: Flickr sharing panel

Given a picture, in fact, Flickr produces a wide set of resized samples, which are also compressed if necessary. In this section we will focus on the following sizes:

- 500 pixels
- 800 pixels *
- Large (which will be 1024 pixels if it exceeds that length)
- 1600 pixels *
- 2048 pixels *
- The original size (original, high resolution file)

How does compression work in all these cases? As before, let's upload both LEP and GNS2C (both produced using Photoshop CS5 with `Save as` compression level 12); then, we re-download and analyse them with `JPEGsnoop`. The results are the following:

Original size

The first thing we notice is that Flickr maintains the additional markers used by Photoshop CS5. The luma-chroma quantization tables are also the same as those used by Photoshop CS5 `Save as` compression level 12, and no chroma subsampling is performed. The Huffman tables are also identical to those used by Photoshop CS5. The resolution and file size are also the same, and a visual comparison between the original samples and the files produced by Flickr shows no differences: in both cases we have no block artefacts, smooth colours, rich shadows and sharp edges. All in all, there is a **perfect correspondence between Flickr "original size" and the sample we uploaded**.

2048 pixels *

First of all, any additional marker used by Photoshop CS5 is discarded to reduce the file size. The luma-chroma quantization tables used in this case have **very high coefficients** and can be compared to those used by Photoshop `Save for web` compression qualities 49-41.

```
DQT, Row #0:   8   5   5   8  12  19  25  29
DQT, Row #1:   6   6   7   9  12  28  29  26
DQT, Row #2:   7   6   8  12  19  28  33  27
DQT, Row #3:   7   8  11  14  25  42  38  30
DQT, Row #4:   8  11  18  27  33  53  49  37
DQT, Row #5:  12  17  26  31  39  50  54  44
DQT, Row #6:  24  31  37  42  49  58  58  49
DQT, Row #7:  35  44  46  47  54  48  49  48

DQT, Row #0:   8   8  12  23  48  48  48  48
DQT, Row #1:   8  10  12  32  48  48  48  48
DQT, Row #2:  12  12  27  48  48  48  48  48
DQT, Row #3:  23  32  48  48  48  48  48  48
DQT, Row #4:  48  48  48  48  48  48  48  48
DQT, Row #5:  48  48  48  48  48  48  48  48
DQT, Row #6:  48  48  48  48  48  48  48  48
DQT, Row #7:  48  48  48  48  48  48  48  48
```

No chroma subsampling is performed and custom Huffman tables with a reduced number of code words are employed[15]. Visually, the sample shows many block artefacts if we zoom at $\approx 1000\%$: critical areas (gradient colours, shadows, small details, ...) do suffer from the compression, showing discolourations and artefacts. Still, there are no posterized areas and the overall quality of the picture is still acceptable.

1600 pixels *

As before, the additional marker inserted by Photoshop CS5 are discarded to reduce the file size. The luma-chroma quantization tables used for this compression are lower than those used in the "2048 pixels" case and can be compared to those used by Photoshop `Save for web` compression qualities 69-67.

```
DQT, Row #0:   4   3   1   4   6   9  12  14
```

[15] See appendix 6 for further details

```
DQT, Row #1:   3   3   3   5   6  14  14  13
DQT, Row #2:   3   3   4   6   9  13  16  13
DQT, Row #3:   3   4   5   7  12  20  18  14
DQT, Row #4:   4   5   9  13  16  25  24  18
DQT, Row #5:   6   8  13  15  19  24  26  21
DQT, Row #6:  11  15  18  20  24  28  28  23
DQT, Row #7:  17  21  22  23  26  23  24  23

DQT, Row #0:   4   4   6  11  23  23  23  23
DQT, Row #1:   4   5   6  15  23  23  23  23
DQT, Row #2:   6   6  13  23  23  23  23  23
DQT, Row #3:  11  15  23  23  23  23  23  23
DQT, Row #4:  23  23  23  23  23  23  23  23
DQT, Row #5:  23  23  23  23  23  23  23  23
DQT, Row #6:  23  23  23  23  23  23  23  23
DQT, Row #7:  23  23  23  23  23  23  23  23
```

No chroma subsampling is performed and custom Huffman tables with a reduced number of code words are employed[16]. Visually, the sample has a limited amount of artefacts, even if we zoom at $\approx 1000\%$: critical areas (gradient colours, shadows, small details, ...) do show some blocks but the overall quality of the samples is more than acceptable.

1024 pixels *

Again, to reduce the file size Flickr discards any additional marker used by Photoshop CS5. The luma-chroma quantization tables used for this compression are a little higher than those used instead for the "1600 pixels" sample, and can be compared to those used by Photoshop `Save for web` compression qualities 64-60.

```
DQT, Row #0:   4   3   3   4   7  11  14  17
DQT, Row #1:   3   3   4   6   7  16  17  15
DQT, Row #2:   4   4   4   7  11  16  20  15
```

[16]See appendix 6 for further details

```
DQT, Row #3:   4   5   6   8  14  24  22  17
DQT, Row #4:   5   6  10  15  19  31  29  21
DQT, Row #5:   7  10  15  18  22  29  32  26
DQT, Row #6:  14  18  22  24  29  34  34  28
DQT, Row #7:  20  26  27  28  31  28  29  28

DQT, Row #0:   5   5   7  13  28  28  28  28
DQT, Row #1:   5   6   7  18  28  28  28  28
DQT, Row #2:   7   7  15  28  28  28  28  28
DQT, Row #3:  13  18  28  28  28  28  28  28
DQT, Row #4:  28  28  28  28  28  28  28  28
DQT, Row #5:  28  28  28  28  28  28  28  28
DQT, Row #6:  28  28  28  28  28  28  28  28
DQT, Row #7:  28  28  28  28  28  28  28  28
```

No chroma subsampling is performed and custom Huffman tables with a reduced number of code words are employed[17]. Visually, the samples show the same amount of block artefacts as before; critical areas have some discolourations but all in all the quality of the samples is acceptable.

800 pixels *

As before, additional markers used by Photoshop CS5 are discarded to reduce the file size. The luma-chroma quantization tables used for this compression are a more or less similar to the previous one, with slightly lower coefficients. They can be compared to those used by Photoshop `Save for web` compression qualities 68-65.

```
DQT, Row #0:   4   3   3   4   6  10  13  15
DQT, Row #1:   3   3   3   5   6  15  15  13
DQT, Row #2:   3   3   4   6  10  14  17  14
DQT, Row #3:   3   4   6   7  13  22  20  15
DQT, Row #4:   4   6   9  14  17  27  25  19
DQT, Row #5:   6   9  13  16  20  26  28  23
```

[17]See appendix 7 for further details

```
DQT, Row #6:  12  16  19  22  25  30  30  25
DQT, Row #7:  18  23  24  24  28  25  25  25

DQT, Row #0:   4   4   6  12  25  25  25  25
DQT, Row #1:   4   5   6  16  25  25  25  25
DQT, Row #2:   6   6  14  25  25  25  25  25
DQT, Row #3:  12  16  25  25  25  25  25  25
DQT, Row #4:  25  25  25  25  25  25  25  25
DQT, Row #5:  25  25  25  25  25  25  25  25
DQT, Row #6:  25  25  25  25  25  25  25  25
DQT, Row #7:  25  25  25  25  25  25  25  25
```

No chroma subsampling is performed and custom Huffman tables with less code words are used[18]. Visually, the samples are generally acceptable but show many block artefacts and discolourations.

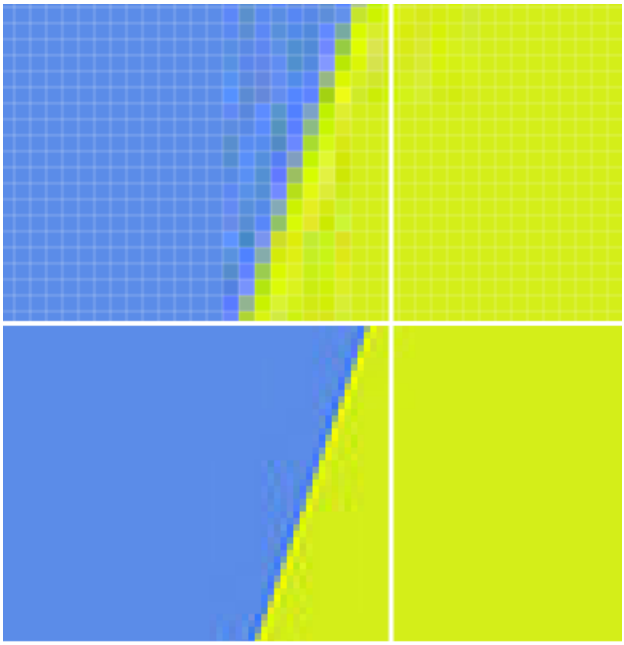

Figure 74: GNS2C compressed by Flickr, 2048px (top) and 800px (bottom)

500 pixels *

[18]See appendix 6 for further details

As before, additional markers used by Photoshop CS5 are discarded to reduce the file size. The luma-chroma quantization tables used for this compression are a more or less similar to the previous one, with slightly lower coefficients. They can be compared to those used by Photoshop `Save for web` compression qualities 68-65.

```
DQT, Row #0:   3   1   1   3   2   7   9  11
DQT, Row #1:   1   1   1   2   2   5  11   9
DQT, Row #2:   1   1   3   2   7   5   6   9
DQT, Row #3:   1   3   2   5   9  15  14  11
DQT, Row #4:   3   2   3   9  12  19  18  13
DQT, Row #5:   2   3   9  11  14  18  20  16
DQT, Row #6:   9  11  13  15  18  21  21  18
DQT, Row #7:  13  16  17  17  20  17  18  17

DQT, Row #0:   3   3   2   4  17  17  17  17
DQT, Row #1:   3   2   2  11  17  17  17  17
DQT, Row #2:   2   2   9  17  17  17  17  17
DQT, Row #3:   4  11  17  17  17  17  17  17
DQT, Row #4:  17  17  17  17  17  17  17  17
DQT, Row #5:  17  17  17  17  17  17  17  17
DQT, Row #6:  17  17  17  17  17  17  17  17
DQT, Row #7:  17  17  17  17  17  17  17  17
```

No chroma subsampling is performed and custom Huffman tables with less code words are used See appendix 6 for further details. Visually, the samples are acceptable but clearly show some artefacts: edges are not sharp, gradient colours show some blocks, smooth surfaces have some discolourations, etc.

To sum up, Flickr performs different compression on the samples depending on the selected size of the file. Despite showing some block artefacts, the results are always more than acceptable and can be easily used on the web in a variety of situations from simple image sharing to building up an online portfolio.

6 Conclusions

In the previous pages we have seen how the JPEG file format is widely used both on the web and by famous raster graphic editors. We have also listed all its positive features and shortcomings, focusing on how compression is performed in several critical situations (edges, shadows, detailed areas, juxtaposed colours, ...). To conclude, the visual and technical analysis of dozens of samples shows that JPEG is an **efficient standard if used wisely**: on average, it performs best with pictorial samples (rich, complex photos) and worst with vector graphics, text or gradient pictures — exactly as expected. Still, JPEG manages to return acceptable or mostly acceptable results even when it is not even supposed to do so, in a manner of speaking: despite its clear limitations, JPEG is a valid compression technique with a lot of potential.

7 Appendix 1 - Photoshop CS5 baseline "Save as"

Compression level: 12

Luminance quantization table:

```
DQT, Row #0:   1   1   1   1   1   1   1   2
DQT, Row #1:   1   1   1   1   1   1   1   2
DQT, Row #2:   1   1   1   1   1   1   2   2
DQT, Row #3:   1   1   1   1   1   2   2   3
DQT, Row #4:   1   1   1   1   2   2   3   3
DQT, Row #5:   1   1   1   2   2   3   3   3
DQT, Row #6:   1   1   2   2   3   3   3   3
DQT, Row #7:   2   2   2   3   3   3   3   3
```

Chrominance quantization table:

```
DQT, Row #0:   1   1   1   2   3   3   3   3
DQT, Row #1:   1   1   1   2   3   3   3   3
DQT, Row #2:   1   1   2   3   3   3   3   3
DQT, Row #3:   2   2   3   3   3   3   3   3
DQT, Row #4:   3   3   3   3   3   3   3   3
DQT, Row #5:   3   3   3   3   3   3   3   3
DQT, Row #6:   3   3   3   3   3   3   3   3
DQT, Row #7:   3   3   3   3   3   3   3   3
```

Chroma subsampling not performed:

```
Component[1]: ... (Subsamp 1 x 1), ... (Lum: Y)
Component[2]: ... (Subsamp 1 x 1), ... (Chrom: Cb)
Component[3]: ... (Subsamp 1 x 1), ... (Chrom: Cr)
```

Huffman tables:

```
Destination ID = 0
Class = 0 (DC / Lossless Table)
  Codes of length 01 bits (000 total):
  Codes of length 02 bits (000 total):
  Codes of length 03 bits (006 total): 07 08 06 05 04 09
  Codes of length 04 bits (002 total): 03 0A
  Codes of length 05 bits (003 total): 02 01 00
  Codes of length 06 bits (001 total): 0B
  Codes of length 07 bits (000 total):
  Codes of length 08 bits (000 total):
  Codes of length 09 bits (000 total):
  Codes of length 10 bits (000 total):
  Codes of length 11 bits (000 total):
  Codes of length 12 bits (000 total):
  Codes of length 13 bits (000 total):
  Codes of length 14 bits (000 total):
  Codes of length 15 bits (000 total):
  Codes of length 16 bits (000 total):
  Total number of codes: 012

----
Destination ID = 1
Class = 0 (DC / Lossless Table)
  Codes of length 01 bits (000 total):
  Codes of length 02 bits (000 total):
  Codes of length 03 bits (006 total): 06 05 04 03 07 02
  Codes of length 04 bits (003 total): 08 01 09
  Codes of length 05 bits (001 total): 00
  Codes of length 06 bits (001 total): 0A
  Codes of length 07 bits (001 total): 0B
  Codes of length 08 bits (000 total):
  Codes of length 09 bits (000 total):
  Codes of length 10 bits (000 total):
  Codes of length 11 bits (000 total):
```

```
  Codes of length 12 bits (000 total):
  Codes of length 13 bits (000 total):
  Codes of length 14 bits (000 total):
  Codes of length 15 bits (000 total):
  Codes of length 16 bits (000 total):
  Total number of codes: 012

----
Destination ID = 0
Class = 1 (AC Table)
  Codes of length 01 bits (000 total):
  Codes of length 02 bits (002 total): 01 02
  Codes of length 03 bits (001 total): 03
  Codes of length 04 bits (003 total): 04 11 05
  Codes of length 05 bits (004 total): 12 06 21 07
  Codes of length 06 bits (001 total): 13
  Codes of length 07 bits (003 total): 22 00 08
  Codes of length 08 bits (003 total): 31 14 41
  Codes of length 09 bits (002 total): 32 23
  Codes of length 10 bits (003 total): 15 09 51
  Codes of length 11 bits (003 total): 42 16 61
  Codes of length 12 bits (003 total): 24 33 17
  Codes of length 13 bits (002 total): 52 71
  Codes of length 14 bits (006 total): 81 18 62 91 25 43
  Codes of length 15 bits (009 total): A1 B1 F0 26 34 72 0A 19 C1
  Codes of length 16 bits (117 total):
      D1 35 27 E1 53 36 82 F1 92 A2 44 54 73 45 46 37
      47 63 28 55 56 57 1A B2 C2 D2 E2 F2 64 83 74 93
      84 65 A3 B3 C3 D3 E3 29 38 66 F3 75 2A 39 3A 48
      49 4A 58 59 5A 67 68 69 6A 76 77 78 79 7A 85 86
      87 88 89 8A 94 95 96 97 98 99 9A A4 A5 A6 A7 A8
      A9 AA B4 B5 B6 B7 B8 B9 BA C4 C5 C6 C7 C8 C9 CA
      D4 D5 D6 D7 D8 D9 DA E4 E5 E6 E7 E8 E9 EA F4 F5
      F6 F7 F8 F9 FA
```

```
  Total number of codes: 162

----
Destination ID = 1
Class = 1 (AC Table)
  Codes of length 01 bits (000 total):
  Codes of length 02 bits (002 total): 01 02
  Codes of length 03 bits (001 total): 03
  Codes of length 04 bits (003 total): 11 04 21
  Codes of length 05 bits (002 total): 12 05
  Codes of length 06 bits (004 total): 31 06 00 22
  Codes of length 07 bits (004 total): 13 41 51 07
  Codes of length 08 bits (003 total): 32 61 14
  Codes of length 09 bits (005 total): 71 08 42 81 23
  Codes of length 10 bits (004 total): 91 15 52 A1
  Codes of length 11 bits (004 total): 62 16 33 09
  Codes of length 12 bits (004 total): B1 24 C1 D1
  Codes of length 13 bits (006 total): 43 72 F0 17 E1 82
  Codes of length 14 bits (006 total): 34 25 92 53 18 63
  Codes of length 15 bits (005 total): 44 F1 A2 B2 26
  Codes of length 16 bits (109 total):
      35 19 54 36 45 64 27 0A 73 83 93 46 74 C2 D2 E2
      F2 55 65 75 56 37 84 85 A3 B3 C3 D3 E3 F3 29 1A
      94 A4 B4 C4 D4 E4 F4 95 A5 B5 C5 D5 E5 F5 28 47
      57 66 38 76 86 96 A6 B6 C6 D6 E6 F6 67 77 87 97
      A7 B7 C7 D7 E7 F7 48 58 68 78 88 98 A8 B8 C8 D8
      E8 F8 39 49 59 69 79 89 99 A9 B9 C9 D9 E9 F9 2A
      3A 4A 5A 6A 7A 8A 9A AA BA CA DA EA FA
  Total number of codes: 162
```

Compression level: 11

Luminance quantization table:

```
  DQT, Row #0:   1   1   1   2   3   3   4   5
```

```
DQT, Row #1:   1   1   1   2   3   4   4   6
DQT, Row #2:   1   1   2   3   4   4   5   7
DQT, Row #3:   2   2   3   4   4   5   7   8
DQT, Row #4:   3   3   4   4   5   7   8   8
DQT, Row #5:   3   4   4   5   7   8   8   8
DQT, Row #6:   4   4   5   7   8   8   8   8
DQT, Row #7:   5   6   7   8   8   8   8   8
```

Chrominance quantization table:

```
DQT, Row #0:   1   2   4   7   8   8   8   8
DQT, Row #1:   2   2   4   7   8   8   8   8
DQT, Row #2:   4   4   7   8   8   8   8   8
DQT, Row #3:   7   7   8   8   8   8   8   8
DQT, Row #4:   8   8   8   8   8   8   8   8
DQT, Row #5:   8   8   8   8   8   8   8   8
DQT, Row #6:   8   8   8   8   8   8   8   8
DQT, Row #7:   8   8   8   8   8   8   8   8
```

Chroma subsampling not performed:

```
Component[1]: ... (Subsamp 1 x 1), ... (Lum: Y)
Component[2]: ... (Subsamp 1 x 1), ... (Chrom: Cb)
Component[3]: ... (Subsamp 1 x 1), ... (Chrom: Cr)
```

Huffman tables:

```
Destination ID = 0
Class = 0 (DC / Lossless Table)
  Codes of length 01 bits (000 total):
  Codes of length 02 bits (000 total):
  Codes of length 03 bits (006 total): 07 08 06 05 04 09
  Codes of length 04 bits (002 total): 03 0A
  Codes of length 05 bits (003 total): 02 01 00
```

```
  Codes of length 06 bits (001 total): 0B
  Codes of length 07 bits (000 total):
  Codes of length 08 bits (000 total):
  Codes of length 09 bits (000 total):
  Codes of length 10 bits (000 total):
  Codes of length 11 bits (000 total):
  Codes of length 12 bits (000 total):
  Codes of length 13 bits (000 total):
  Codes of length 14 bits (000 total):
  Codes of length 15 bits (000 total):
  Codes of length 16 bits (000 total):
  Total number of codes: 012

----
Destination ID = 1
Class = 0 (DC / Lossless Table)
  Codes of length 01 bits (000 total):
  Codes of length 02 bits (000 total):
  Codes of length 03 bits (006 total): 06 05 04 03 07 02
  Codes of length 04 bits (003 total): 08 01 09
  Codes of length 05 bits (001 total): 00
  Codes of length 06 bits (001 total): 0A
  Codes of length 07 bits (001 total): 0B
  Codes of length 08 bits (000 total):
  Codes of length 09 bits (000 total):
  Codes of length 10 bits (000 total):
  Codes of length 11 bits (000 total):
  Codes of length 12 bits (000 total):
  Codes of length 13 bits (000 total):
  Codes of length 14 bits (000 total):
  Codes of length 15 bits (000 total):
  Codes of length 16 bits (000 total):
  Total number of codes: 012
```

```
----
Destination ID = 0
Class = 1 (AC Table)
  Codes of length 01 bits (000 total):
  Codes of length 02 bits (002 total): 01 02
  Codes of length 03 bits (001 total): 03
  Codes of length 04 bits (002 total): 04 11
  Codes of length 05 bits (005 total): 05 06 21 12 00
  Codes of length 06 bits (002 total): 07 31
  Codes of length 07 bits (003 total): 41 13 08
  Codes of length 08 bits (004 total): 51 22 61 14
  Codes of length 09 bits (006 total): 71 81 32 91 09 A1
  Codes of length 10 bits (006 total): 23 F0 C1 42 B1 15
  Codes of length 11 bits (005 total): D1 16 E1 F1 52
  Codes of length 12 bits (005 total): 33 17 24 62 18
  Codes of length 13 bits (001 total): 43
  Codes of length 14 bits (003 total): 34 25 82
  Codes of length 15 bits (006 total): 0A 19 72 53 26 63
  Codes of length 16 bits (111 total):
      92 44 35 A2 54 B2 1A 73 36 C2 D2 27 45 37 46 E2
      F2 83 93 A3 B3 64 55 28 C3 D3 29 38 E3 F3 47 48
      56 65 2A 39 3A 49 4A 57 58 59 5A 66 74 75 84 85
      67 76 77 68 86 87 94 95 A4 A5 B4 B5 C4 C5 D4 D5
      E4 E5 F4 F5 96 97 A6 A7 B6 B7 C6 C7 D6 D7 E6 E7
      F6 F7 69 6A 78 79 7A 88 89 8A 98 99 9A A8 A9 AA
      B8 B9 BA C8 C9 CA D8 D9 DA E8 E9 EA F8 F9 FA
  Total number of codes: 162

----
Destination ID = 1
Class = 1 (AC Table)
  Codes of length 01 bits (000 total):
  Codes of length 02 bits (001 total): 01
  Codes of length 03 bits (003 total): 02 03 11
```

```
Codes of length 04 bits (002 total): 00 04
Codes of length 05 bits (003 total): 21 05 12
Codes of length 06 bits (004 total): 31 06 41 F0
Codes of length 07 bits (007 total): 51 61 07 13 22 71 81
Codes of length 08 bits (006 total): 91 A1 B1 C1 08 32
Codes of length 09 bits (003 total): D1 14 E1
Codes of length 10 bits (004 total): 23 F1 42 15
Codes of length 11 bits (003 total): 52 09 16
Codes of length 12 bits (006 total): 33 62 D2 72 24 82
Codes of length 13 bits (007 total): C2 92 93 43 17 73 83
Codes of length 14 bits (007 total): A2 B2 63 25 34 53 E2
Codes of length 15 bits (001 total): B3
Codes of length 16 bits (105 total):
    35 26 44 54 64 45 55 27 0A 84 B4 18 19 1A 28 29
    2A 36 37 38 39 3A 46 47 48 49 4A 56 57 58 59 5A
    65 66 67 68 69 6A 74 75 76 77 78 79 7A 85 86 87
    88 89 8A 94 95 96 97 98 99 9A A3 A4 A5 A6 A7 A8
    A9 AA B5 B6 B7 B8 B9 BA C3 C4 C5 C6 C7 C8 C9 CA
    D3 D4 D5 D6 D7 D8 D9 DA E3 E4 E5 E6 E7 E8 E9 EA
    F2 F3 F4 F5 F6 F7 F8 F9 FA
Total number of codes: 162
```

Compression level: 10

Luminance quantization table:

```
DQT, Row #0:   2   2   3   4   5   6   8  11
DQT, Row #1:   2   2   2   4   5   7   9  11
DQT, Row #2:   3   2   3   5   7   9  11  12
DQT, Row #3:   4   4   5   7   9  11  12  12
DQT, Row #4:   5   5   7   9  11  12  12  12
DQT, Row #5:   6   7   9  11  12  12  12  12
DQT, Row #6:   8   9  11  12  12  12  12  12
DQT, Row #7:  11  11  12  12  12  12  12  12
```

Chrominance quantization table:

```
DQT, Row #0:   3   3   7  13  15  15  15  15
DQT, Row #1:   3   4   7  13  14  12  12  12
DQT, Row #2:   7   7  13  14  12  12  12  12
DQT, Row #3:  13  13  14  12  12  12  12  12
DQT, Row #4:  15  14  12  12  12  12  12  12
DQT, Row #5:  15  12  12  12  12  12  12  12
DQT, Row #6:  15  12  12  12  12  12  12  12
DQT, Row #7:  15  12  12  12  12  12  12  12
```

Chroma subsampling not performed:

```
Component[1]: ... (Subsamp 1 x 1), ... (Lum: Y)
Component[2]: ... (Subsamp 1 x 1), ... (Chrom: Cb)
Component[3]: ... (Subsamp 1 x 1), ... (Chrom: Cr)
```

Huffman tables:

```
  Destination ID = 0
  Class = 0 (DC / Lossless Table)
    Codes of length 01 bits (000 total):
    Codes of length 02 bits (000 total):
    Codes of length 03 bits (007 total): 04 05 03 02 06 01 00
    Codes of length 04 bits (001 total): 07
    Codes of length 05 bits (001 total): 08
    Codes of length 06 bits (001 total): 09
    Codes of length 07 bits (001 total): 0A
    Codes of length 08 bits (001 total): 0B
    Codes of length 09 bits (000 total):
    Codes of length 10 bits (000 total):
    Codes of length 11 bits (000 total):
    Codes of length 12 bits (000 total):
    Codes of length 13 bits (000 total):
    Codes of length 14 bits (000 total):
    Codes of length 15 bits (000 total):
```

```
  Codes of length 16 bits (000 total):
  Total number of codes: 012

----
Destination ID = 1
Class = 0 (DC / Lossless Table)
  Codes of length 01 bits (000 total):
  Codes of length 02 bits (002 total): 01 00
  Codes of length 03 bits (002 total): 02 03
  Codes of length 04 bits (003 total): 04 05 06
  Codes of length 05 bits (001 total): 07
  Codes of length 06 bits (001 total): 08
  Codes of length 07 bits (001 total): 09
  Codes of length 08 bits (001 total): 0A
  Codes of length 09 bits (001 total): 0B
  Codes of length 10 bits (000 total):
  Codes of length 11 bits (000 total):
  Codes of length 12 bits (000 total):
  Codes of length 13 bits (000 total):
  Codes of length 14 bits (000 total):
  Codes of length 15 bits (000 total):
  Codes of length 16 bits (000 total):
  Total number of codes: 012

----
Destination ID = 0
Class = 1 (AC Table)
  Codes of length 01 bits (000 total):
  Codes of length 02 bits (002 total): 01 02
  Codes of length 03 bits (001 total): 03
  Codes of length 04 bits (003 total): 11 04 00
  Codes of length 05 bits (003 total): 05 21 12
  Codes of length 06 bits (002 total): 31 41
  Codes of length 07 bits (004 total): 51 06 13 61
```

```
  Codes of length 08 bits (002 total): 22 71
  Codes of length 09 bits (006 total): 81 14 32 91 A1 07
  Codes of length 10 bits (007 total): 15 B1 42 23 C1 52 D1
  Codes of length 11 bits (003 total): E1 33 16
  Codes of length 12 bits (004 total): 62 F0 24 72
  Codes of length 13 bits (002 total): 82 F1
  Codes of length 14 bits (006 total): 25 43 34 53 92 A2
  Codes of length 15 bits (002 total): B2 63
  Codes of length 16 bits (115 total):
      73 C2 35 44 27 93 A3 B3 36 17 54 64 74 C3 D2 E2
      08 26 83 09 0A 18 19 84 94 45 46 A4 B4 56 D3 55
      28 1A F2 E3 F3 C4 D4 E4 F4 65 75 85 95 A5 B5 C5
      D5 E5 F5 66 76 86 96 A6 B6 C6 D6 E6 F6 37 47 57
      67 77 87 97 A7 B7 C7 D7 E7 F7 38 48 58 68 78 88
      98 A8 B8 C8 D8 E8 F8 29 39 49 59 69 79 89 99 A9
      B9 C9 D9 E9 F9 2A 3A 4A 5A 6A 7A 8A 9A AA BA CA
      DA EA FA
  Total number of codes: 162

----
Destination ID = 1
Class = 1 (AC Table)
  Codes of length 01 bits (000 total):
  Codes of length 02 bits (002 total): 01 00
  Codes of length 03 bits (002 total): 02 11
  Codes of length 04 bits (001 total): 03
  Codes of length 05 bits (002 total): 04 21
  Codes of length 06 bits (003 total): 12 31 41
  Codes of length 07 bits (005 total): 05 51 13 61 22
  Codes of length 08 bits (005 total): 06 71 81 91 32
  Codes of length 09 bits (004 total): A1 B1 F0 14
  Codes of length 10 bits (005 total): C1 D1 E1 23 42
  Codes of length 11 bits (006 total): 15 52 62 72 F1 33
  Codes of length 12 bits (004 total): 24 34 43 82
```

```
Codes of length 13 bits (008 total): 16 92 53 25 A2 63 B2 C2
Codes of length 14 bits (003 total): 07 73 D2
Codes of length 15 bits (003 total): 35 E2 44
Codes of length 16 bits (109 total):
    83 17 54 93 08 09 0A 18 19 26 36 45 1A 27 64 74
    55 37 F2 A3 B3 C3 28 29 D3 E3 F3 84 94 A4 B4 C4
    D4 E4 F4 65 75 85 95 A5 B5 C5 D5 E5 F5 46 56 66
    76 86 96 A6 B6 C6 D6 E6 F6 47 57 67 77 87 97 A7
    B7 C7 D7 E7 F7 38 48 58 68 78 88 98 A8 B8 C8 D8
    E8 F8 39 49 59 69 79 89 99 A9 B9 C9 D9 E9 F9 2A
    3A 4A 5A 6A 7A 8A 9A AA BA CA DA EA FA
Total number of codes: 162
```

Compression level: 9

Luminance quantization table:

```
DQT, Row #0:   4   3   4   7   9  11  14  17
DQT, Row #1:   3   3   4   7   9  12  12  12
DQT, Row #2:   4   4   5   9  12  12  12  12
DQT, Row #3:   7   7   9  12  12  12  12  12
DQT, Row #4:   9   9  12  12  12  12  12  12
DQT, Row #5:  11  12  12  12  12  12  12  12
DQT, Row #6:  14  12  12  12  12  12  12  12
DQT, Row #7:  17  12  12  12  12  12  12  12
```

Chrominance quantization table:

```
DQT, Row #0:   4   6  12  22  20  20  17  17
DQT, Row #1:   6   8  12  14  14  12  12  12
DQT, Row #2:  12  12  14  14  12  12  12  12
DQT, Row #3:  22  14  14  12  12  12  12  12
DQT, Row #4:  20  14  12  12  12  12  12  12
DQT, Row #5:  20  12  12  12  12  12  12  12
DQT, Row #6:  17  12  12  12  12  12  12  12
DQT, Row #7:  17  12  12  12  12  12  12  12
```

Chroma subsampling not performed:

```
Component[1]: ... (Subsamp 1 x 1), ... (Lum: Y)
Component[2]: ... (Subsamp 1 x 1), ... (Chrom: Cb)
Component[3]: ... (Subsamp 1 x 1), ... (Chrom: Cr)
```

Huffman table: same as compression level 10.

Compression level: 8

Luminance quantization table:

```
DQT, Row #0:   6   4   7  11  14  17  22  17
DQT, Row #1:   4   5   6  10  14  19  12  12
DQT, Row #2:   7   6   8  14  19  12  12  12
DQT, Row #3:  11  10  14  19  12  12  12  12
DQT, Row #4:  14  14  19  12  12  12  12  12
DQT, Row #5:  17  19  12  12  12  12  12  12
DQT, Row #6:  22  12  12  12  12  12  12  12
DQT, Row #7:  17  12  12  12  12  12  12  12
```

Chrominance quantization table:

```
DQT, Row #0:   7   9  19  34  20  20  17  17
DQT, Row #1:   9  12  19  14  14  12  12  12
DQT, Row #2:  19  19  14  14  12  12  12  12
DQT, Row #3:  34  14  14  12  12  12  12  12
DQT, Row #4:  20  14  12  12  12  12  12  12
DQT, Row #5:  20  12  12  12  12  12  12  12
DQT, Row #6:  17  12  12  12  12  12  12  12
DQT, Row #7:  17  12  12  12  12  12  12  12
```

Chroma subsampling not performed:

```
Component[1]: ... (Subsamp 1 x 1), ... (Lum: Y)
Component[2]: ... (Subsamp 1 x 1), ... (Chrom: Cb)
Component[3]: ... (Subsamp 1 x 1), ... (Chrom: Cr)
```

Huffman table: same as compression level 10.

Compression level: 7

Luminance quantization table:

```
DQT, Row #0:  10   7  11  18  22  27  34  17
DQT, Row #1:   7   8  10  17  23  23  12  12
DQT, Row #2:  11  10  14  22  23  12  12  12
DQT, Row #3:  18  17  22  23  12  12  12  12
DQT, Row #4:  22  23  23  12  12  12  12  12
DQT, Row #5:  27  23  12  12  12  12  12  12
DQT, Row #6:  34  12  12  12  12  12  12  12
DQT, Row #7:  17  12  12  12  12  12  12  12
```

Chrominance quantization table:

```
DQT, Row #0:  11  14  31  34  20  20  17  17
DQT, Row #1:  14  19  24  14  14  12  12  12
DQT, Row #2:  31  24  14  14  12  12  12  12
DQT, Row #3:  34  14  14  12  12  12  12  12
DQT, Row #4:  20  14  12  12  12  12  12  12
DQT, Row #5:  20  12  12  12  12  12  12  12
DQT, Row #6:  17  12  12  12  12  12  12  12
DQT, Row #7:  17  12  12  12  12  12  12  12
```

Chroma subsampling not performed:

```
Component[1]: ... (Subsamp 1 x 1), ... (Lum: Y)
Component[2]: ... (Subsamp 1 x 1), ... (Chrom: Cb)
Component[3]: ... (Subsamp 1 x 1), ... (Chrom: Cr)
```

Huffman table: same as compression level 10.

Compression level: 6

Luminance quantization table:

```
DQT, Row #0:   8   6   9  14  17  21  28  17
DQT, Row #1:   6   6   8  13  18  23  12  12
DQT, Row #2:   9   8  11  17  23  12  12  12
DQT, Row #3:  14  13  17  23  12  12  12  12
DQT, Row #4:  17  18  23  12  12  12  12  12
DQT, Row #5:  21  23  12  12  12  12  12  12
DQT, Row #6:  28  12  12  12  12  12  12  12
DQT, Row #7:  17  12  12  12  12  12  12  12
```

Chrominance quantization table:

```
DQT, Row #0:   9   9  11  18  20  20  17  17
DQT, Row #1:   9  10  11  14  14  12  12  12
DQT, Row #2:  11  11  14  14  12  12  12  12
DQT, Row #3:  18  14  14  12  12  12  12  12
DQT, Row #4:  20  14  12  12  12  12  12  12
DQT, Row #5:  20  12  12  12  12  12  12  12
DQT, Row #6:  17  12  12  12  12  12  12  12
DQT, Row #7:  17  12  12  12  12  12  12  12
```

Chroma subsampling performed:

```
Component[1]: ... (Subsamp 1 x 1), ... (Lum: Y)
Component[2]: ... (Subsamp 2 x 2), ... (Chrom: Cb)
Component[3]: ... (Subsamp 2 x 2), ... (Chrom: Cr)
```

Huffman table: same as compression level 10.

Compression level: 5

Luminance quantization table:

```
DQT, Row #0:  12   8  13  21  26  32  34  17
DQT, Row #1:   8   9  12  20  27  23  12  12
DQT, Row #2:  13  12  16  26  23  12  12  12
```

```
DQT, Row #3:  21  20  26  23  12  12  12  12
DQT, Row #4:  26  27  23  12  12  12  12  12
DQT, Row #5:  32  23  12  12  12  12  12  12
DQT, Row #6:  34  12  12  12  12  12  12  12
DQT, Row #7:  17  12  12  12  12  12  12  1
```

Chrominance quantization table:

```
DQT, Row #0:  13  13  17  27  20  20  17  17
DQT, Row #1:  13  14  17  14  14  12  12  12
DQT, Row #2:  17  17  14  14  12  12  12  12
DQT, Row #3:  27  14  14  12  12  12  12  12
DQT, Row #4:  20  14  12  12  12  12  12  12
DQT, Row #5:  20  12  12  12  12  12  12  12
DQT, Row #6:  17  12  12  12  12  12  12  12
DQT, Row #7:  17  12  12  12  12  12  12  12
```

Chroma subsampling performed:

```
Component[1]: ... (Subsamp 1 x 1), ... (Lum: Y)
Component[2]: ... (Subsamp 2 x 2), ... (Chrom: Cb)
Component[3]: ... (Subsamp 2 x 2), ... (Chrom: Cr)
```

Huffman table:

```
Destination ID = 0
Class = 0 (DC / Lossless Table)
  Codes of length 01 bits (000 total):
  Codes of length 02 bits (001 total): 03
  Codes of length 03 bits (005 total): 00 01 02 04 05
  Codes of length 04 bits (001 total): 06
  Codes of length 05 bits (001 total): 07
  Codes of length 06 bits (001 total): 08
  Codes of length 07 bits (001 total): 09
```

```
  Codes of length 08 bits (001 total): 0A
  Codes of length 09 bits (001 total): 0B
  Codes of length 10 bits (000 total):
  Codes of length 11 bits (000 total):
  Codes of length 12 bits (000 total):
  Codes of length 13 bits (000 total):
  Codes of length 14 bits (000 total):
  Codes of length 15 bits (000 total):
  Codes of length 16 bits (000 total):
  Total number of codes: 012

----
Destination ID = 1
Class = 0 (DC / Lossless Table)
  Codes of length 01 bits (000 total):
  Codes of length 02 bits (001 total): 01
  Codes of length 03 bits (005 total): 00 02 03 04 05
  Codes of length 04 bits (001 total): 06
  Codes of length 05 bits (001 total): 07
  Codes of length 06 bits (001 total): 08
  Codes of length 07 bits (001 total): 09
  Codes of length 08 bits (001 total): 0A
  Codes of length 09 bits (001 total): 0B
  Codes of length 10 bits (000 total):
  Codes of length 11 bits (000 total):
  Codes of length 12 bits (000 total):
  Codes of length 13 bits (000 total):
  Codes of length 14 bits (000 total):
  Codes of length 15 bits (000 total):
  Codes of length 16 bits (000 total):
  Total number of codes: 012

----
Destination ID = 0
```

```
Class = 1 (AC Table)
  Codes of length 01 bits (000 total):
  Codes of length 02 bits (001 total): 01
  Codes of length 03 bits (004 total): 00 02 11 03
  Codes of length 04 bits (001 total): 04
  Codes of length 05 bits (003 total): 21 12 31
  Codes of length 06 bits (002 total): 05 41
  Codes of length 07 bits (004 total): 51 61 13 22
  Codes of length 08 bits (002 total): 71 81
  Codes of length 09 bits (005 total): 32 06 14 91 A1
  Codes of length 10 bits (007 total): B1 42 23 24 15 52 C1
  Codes of length 11 bits (006 total): 62 33 34 72 82 D1
  Codes of length 12 bits (008 total): 43 07 25 92 53 F0 E1 F1
  Codes of length 13 bits (005 total): 63 73 35 16 A2
  Codes of length 14 bits (003 total): B2 83 26
  Codes of length 15 bits (012 total): 44 93 54 64 45 C2 A3 74
                                       36 17 D2 55
  Codes of length 16 bits (051 total):
      E2 65 F2 B3 84 C3 D3 75 E3 F3 46 27 94 A4 85 B4
      95 C4 D4 E4 F4 A5 B5 C5 D5 E5 F5 56 66 76 86 96
      A6 B6 C6 D6 E6 F6 37 47 57 67 77 87 97 A7 B7 C7
      D7 E7 F7
  Total number of codes: 114

----
Destination ID = 1
Class = 1 (AC Table)
  Codes of length 01 bits (000 total):
  Codes of length 02 bits (002 total): 01 00
  Codes of length 03 bits (002 total): 02 11
  Codes of length 04 bits (001 total): 03
  Codes of length 05 bits (002 total): 21 31
  Codes of length 06 bits (004 total): 12 04 41 51
  Codes of length 07 bits (004 total): 61 71 22 13
```

```
Codes of length 08 bits (003 total): 05 32 81
Codes of length 09 bits (004 total): 91 14 A1 B1
Codes of length 10 bits (005 total): 42 23 C1 52 D1
Codes of length 11 bits (006 total): F0 33 24 62 E1 72
Codes of length 12 bits (007 total): 82 92 43 53 15 63 73
Codes of length 13 bits (007 total): 34 F1 25 06 16 A2 B2
Codes of length 14 bits (006 total): 83 07 26 35 C2 D2
Codes of length 15 bits (005 total): 44 93 54 A3 17
Codes of length 16 bits (053 total):
    64 45 55 36 74 65 E2 F2 B3 84 C3 D3 75 E3 F3 46
    94 A4 85 B4 95 C4 D4 E4 F4 A5 B5 C5 D5 E5 F5 56
    66 76 86 96 A6 B6 C6 D6 E6 F6 27 37 47 57 67 77
    87 97 A7 B7 C7
Total number of codes: 111
```

Compression level: 4

Luminance quantization table:

```
DQT, Row #0:  16  11  17  27  34  39  34  17
DQT, Row #1:  11  12  16  26  28  23  12  12
DQT, Row #2:  17  16  21  28  23  12  12  12
DQT, Row #3:  27  26  28  23  12  12  12  12
DQT, Row #4:  34  28  23  12  12  12  12  12
DQT, Row #5:  39  23  12  12  12  12  12  12
DQT, Row #6:  34  12  12  12  12  12  12  12
DQT, Row #7:  17  12  12  12  12  12  12  12
```

Chrominance quantization table:

```
DQT, Row #0:  17  17  22  34  20  20  17  17
DQT, Row #1:  17  19  22  14  14  12  12  12
DQT, Row #2:  22  22  14  14  12  12  12  12
DQT, Row #3:  34  14  14  12  12  12  12  12
DQT, Row #4:  20  14  12  12  12  12  12  12
```

```
DQT, Row #5:  20  12  12  12  12  12  12  12
DQT, Row #6:  17  12  12  12  12  12  12  12
DQT, Row #7:  17  12  12  12  12  12  12  12
```

Chroma subsampling performed:

```
Component[1]: ... (Subsamp 1 x 1), ... (Lum: Y)
Component[2]: ... (Subsamp 2 x 2), ... (Chrom: Cb)
Component[3]: ... (Subsamp 2 x 2), ... (Chrom: Cr)
```

Huffman tables: same as compression level 5.

Compression level: 3

Luminance quantization table:

```
DQT, Row #0:  18  14  22  35  44  39  34  17
DQT, Row #1:  14  16  21  34  28  23  12  12
DQT, Row #2:  22  21  27  28  23  12  12  12
DQT, Row #3:  35  34  28  23  12  12  12  12
DQT, Row #4:  44  28  23  12  12  12  12  12
DQT, Row #5:  39  23  12  12  12  12  12  12
DQT, Row #6:  34  12  12  12  12  12  12  12
DQT, Row #7:  17  12  12  12  12  12  12  12
```

Chrominance quantization table:

```
DQT, Row #0:  20  22  29  34  20  20  17  17
DQT, Row #1:  22  25  24  14  14  12  12  12
DQT, Row #2:  29  24  14  14  12  12  12  12
DQT, Row #3:  34  14  14  12  12  12  12  12
DQT, Row #4:  20  14  12  12  12  12  12  12
DQT, Row #5:  20  12  12  12  12  12  12  12
DQT, Row #6:  17  12  12  12  12  12  12  12
DQT, Row #7:  17  12  12  12  12  12  12  12
```

Chroma subsampling performed:

```
Component[1]: ... (Subsamp 1 x 1), ... (Lum: Y)
Component[2]: ... (Subsamp 2 x 2), ... (Chrom: Cb)
Component[3]: ... (Subsamp 2 x 2), ... (Chrom: Cr)
```

Huffman tables: same as compression level 5.

Compression level: 2

Luminance quantization table:

```
DQT, Row #0:  20  17  26  41  51  39  34  17
DQT, Row #1:  17  18  24  39  28  23  12  12
DQT, Row #2:  26  24  32  28  23  12  12  12
DQT, Row #3:  41  39  28  23  12  12  12  12
DQT, Row #4:  51  28  23  12  12  12  12  12
DQT, Row #5:  39  23  12  12  12  12  12  12
DQT, Row #6:  34  12  12  12  12  12  12  12
DQT, Row #7:  17  12  12  12  12  12  12  12
```

Chrominance quantization table:

```
DQT, Row #0:  21  26  33  34  20  20  17  17
DQT, Row #1:  26  29  24  14  14  12  12  12
DQT, Row #2:  33  24  14  14  12  12  12  12
DQT, Row #3:  34  14  14  12  12  12  12  12
DQT, Row #4:  20  14  12  12  12  12  12  12
DQT, Row #5:  20  12  12  12  12  12  12  12
DQT, Row #6:  17  12  12  12  12  12  12  12
DQT, Row #7:  17  12  12  12  12  12  12  12
```

Chroma subsampling performed:

```
Component[1]: ... (Subsamp 1 x 1), ... (Lum: Y)
Component[2]: ... (Subsamp 2 x 2), ... (Chrom: Cb)
Component[3]: ... (Subsamp 2 x 2), ... (Chrom: Cr)
```

Huffman tables:

```
Destination ID = 0
Class = 0 (DC / Lossless Table)
  Codes of length 01 bits (000 total):
  Codes of length 02 bits (003 total): 01 00 02
  Codes of length 03 bits (001 total): 03
  Codes of length 04 bits (001 total): 04
  Codes of length 05 bits (001 total): 05
  Codes of length 06 bits (001 total): 06
  Codes of length 07 bits (001 total): 07
  Codes of length 08 bits (001 total): 08
  Codes of length 09 bits (001 total): 09
  Codes of length 10 bits (001 total): 0A
  Codes of length 11 bits (001 total): 0B
  Codes of length 12 bits (000 total):
  Codes of length 13 bits (000 total):
  Codes of length 14 bits (000 total):
  Codes of length 15 bits (000 total):
  Codes of length 16 bits (000 total):
  Total number of codes: 012

----
Destination ID = 1
Class = 0 (DC / Lossless Table)
  Codes of length 01 bits (001 total): 00
  Codes of length 02 bits (001 total): 01
  Codes of length 03 bits (001 total): 02
  Codes of length 04 bits (001 total): 03
  Codes of length 05 bits (001 total): 04
  Codes of length 06 bits (001 total): 05
  Codes of length 07 bits (001 total): 06
  Codes of length 08 bits (001 total): 07
  Codes of length 09 bits (001 total): 08
```

```
  Codes of length 10 bits (001 total): 09
  Codes of length 11 bits (001 total): 0A
  Codes of length 12 bits (001 total): 0B
  Codes of length 13 bits (000 total):
  Codes of length 14 bits (000 total):
  Codes of length 15 bits (000 total):
  Codes of length 16 bits (000 total):
  Total number of codes: 012

----
Destination ID = 0
Class = 1 (AC Table)
  Codes of length 01 bits (000 total):
  Codes of length 02 bits (002 total): 01 00
  Codes of length 03 bits (002 total): 02 11
  Codes of length 04 bits (001 total): 03
  Codes of length 05 bits (003 total): 21 12 31
  Codes of length 06 bits (002 total): 04 41
  Codes of length 07 bits (003 total): 51 22 13
  Codes of length 08 bits (004 total): 61 71 32 81
  Codes of length 09 bits (007 total): 91 B1 42 A1 05 D1 C1
  Codes of length 10 bits (006 total): 14 F0 52 23 72 33
  Codes of length 11 bits (003 total): 62 E1 82
  Codes of length 12 bits (003 total): F1 43 34
  Codes of length 13 bits (006 total): 92 A2 B2 15 D2 53
  Codes of length 14 bits (002 total): 24 73
  Codes of length 15 bits (001 total): C2
  Codes of length 16 bits (053 total):
      63 06 83 93 E2 F2 A3 44 54 64 25 35 45 16 26 74
      36 55 65 B3 84 C3 D3 75 E3 F3 46 94 A4 85 B4 95
      C4 D4 E4 F4 A5 B5 C5 D5 E5 F5 56 66 76 86 96 A6
      B6 C6 D6 E6 F6
  Total number of codes: 098
```

```
----
Destination ID = 1
Class = 1 (AC Table)
  Codes of length 01 bits (000 total):
  Codes of length 02 bits (002 total): 00 01
  Codes of length 03 bits (002 total): 11 02
  Codes of length 04 bits (000 total):
  Codes of length 05 bits (005 total): 21 03 31 41 12
  Codes of length 06 bits (001 total): 51
  Codes of length 07 bits (006 total): 61 71 81 91 22 13
  Codes of length 08 bits (006 total): 32 F0 A1 B1 04 C1
  Codes of length 09 bits (001 total): D1
  Codes of length 10 bits (003 total): E1 F1 42
  Codes of length 11 bits (001 total): 52
  Codes of length 12 bits (003 total): 23 62 72
  Codes of length 13 bits (005 total): 14 92 33 82 43
  Codes of length 14 bits (003 total): 24 A2 B2
  Codes of length 15 bits (006 total): 34 53 44 63 73 C2
  Codes of length 16 bits (047 total):
      D2 83 93 A3 54 E2 F2 05 15 25 06 16 26 35 64 45
      55 36 74 65 B3 84 C3 D3 75 E3 F3 46 94 A4 85 B4
      95 C4 D4 E4 F4 A5 B5 C5 D5 E5 F5 56 66 76 86
  Total number of codes: 091
```

Compression level: 1

Luminance quantization table:

```
  DQT, Row #0:  27  26  41  65  66  39  34  17
  DQT, Row #1:  26  29  38  47  28  23  12  12
  DQT, Row #2:  41  38  47  28  23  12  12  12
  DQT, Row #3:  65  47  28  23  12  12  12  12
  DQT, Row #4:  66  28  23  12  12  12  12  12
  DQT, Row #5:  39  23  12  12  12  12  12  12
```

```
DQT, Row #6:  34  12  12  12  12  12  12  12
DQT, Row #7:  17  12  12  12  12  12  12  12
```

Chrominance quantization table:

```
DQT, Row #0:  29  41  52  34  20  20  17  17
DQT, Row #1:  41  38  24  14  14  12  12  12
DQT, Row #2:  52  24  14  14  12  12  12  12
DQT, Row #3:  34  14  14  12  12  12  12  12
DQT, Row #4:  20  14  12  12  12  12  12  12
DQT, Row #5:  20  12  12  12  12  12  12  12
DQT, Row #6:  17  12  12  12  12  12  12  12
DQT, Row #7:  17  12  12  12  12  12  12  12
```

Chroma subsampling performed:

```
Component[1]: ... (Subsamp 1 x 1), ... (Lum: Y)
Component[2]: ... (Subsamp 2 x 2), ... (Chrom: Cb)
Component[3]: ... (Subsamp 2 x 2), ... (Chrom: Cr)
```

Huffman tables: same as compression level 2;

Compression level: 0

Luminance quantization table:

```
DQT, Row #0:  32  33  51  81  66  39  34  17
DQT, Row #1:  33  36  48  47  28  23  12  12
DQT, Row #2:  51  48  47  28  23  12  12  12
DQT, Row #3:  81  47  28  23  12  12  12  12
DQT, Row #4:  66  28  23  12  12  12  12  12
DQT, Row #5:  39  23  12  12  12  12  12  12
DQT, Row #6:  34  12  12  12  12  12  12  12
DQT, Row #7:  17  12  12  12  12  12  12  12
```

Chrominance quantization table:

```
DQT, Row #0:  34  51  52  34  20  20  17  17
DQT, Row #1:  51  38  24  14  14  12  12  12
DQT, Row #2:  52  24  14  14  12  12  12  12
DQT, Row #3:  34  14  14  12  12  12  12  12
DQT, Row #4:  20  14  12  12  12  12  12  12
DQT, Row #5:  20  12  12  12  12  12  12  12
DQT, Row #6:  17  12  12  12  12  12  12  12
DQT, Row #7:  17  12  12  12  12  12  12  12
```

Chroma subsampling performed:

```
Component[1]: ... (Subsamp 1 x 1), ... (Lum: Y)
Component[2]: ... (Subsamp 2 x 2), ... (Chrom: Cb)
Component[3]: ... (Subsamp 2 x 2), ... (Chrom: Cr)
```

Huffman tables: same as compression level 2.

8 Appendix 2 - Photoshop CS5 baseline "Save for web"

Compression quality: 100

Luminance quantization table:

```
DQT, Row #0:   1   1   1   1   1   1   1   1
DQT, Row #1:   1   1   1   1   1   1   1   1
DQT, Row #2:   1   1   1   1   1   1   1   2
DQT, Row #3:   1   1   1   1   1   1   2   2
DQT, Row #4:   1   1   1   1   1   2   2   3
DQT, Row #5:   1   1   1   1   2   2   3   3
DQT, Row #6:   1   1   1   2   2   3   3   3
DQT, Row #7:   1   1   2   2   3   3   3   3
```

Chrominance quantization table:

```
DQT, Row #0:   1   1   1   2   2   3   3   3
DQT, Row #1:   1   1   1   2   3   3   3   3
DQT, Row #2:   1   1   1   3   3   3   3   3
DQT, Row #3:   2   2   3   3   3   3   3   3
DQT, Row #4:   2   3   3   3   3   3   3   3
DQT, Row #5:   3   3   3   3   3   3   3   3
DQT, Row #6:   3   3   3   3   3   3   3   3
DQT, Row #7:   3   3   3   3   3   3   3   3
```

Chroma subsampling not performed:

```
Component[1]: ... (Subsamp 1 x 1), ... (Lum: Y)
Component[2]: ... (Subsamp 1 x 1), ... (Chrom: Cb)
Component[3]: ... (Subsamp 1 x 1), ... (Chrom: Cr)
```

Huffman tables:

```
  Destination ID = 0
  Class = 0 (DC / Lossless Table)
    Codes of length 01 bits (000 total):
    Codes of length 02 bits (000 total):
    Codes of length 03 bits (006 total): 07 08 06 05 04 09
    Codes of length 04 bits (002 total): 03 0A
    Codes of length 05 bits (003 total): 02 01 00
    Codes of length 06 bits (001 total): 0B
    Codes of length 07 bits (000 total):
    Codes of length 08 bits (000 total):
    Codes of length 09 bits (000 total):
    Codes of length 10 bits (000 total):
    Codes of length 11 bits (000 total):
    Codes of length 12 bits (000 total):
    Codes of length 13 bits (000 total):
    Codes of length 14 bits (000 total):
    Codes of length 15 bits (000 total):
```

```
  Codes of length 16 bits (000 total):
  Total number of codes: 012

----
Destination ID = 1
Class = 0 (DC / Lossless Table)
  Codes of length 01 bits (000 total):
  Codes of length 02 bits (000 total):
  Codes of length 03 bits (006 total): 06 05 04 03 07 02
  Codes of length 04 bits (003 total): 08 01 09
  Codes of length 05 bits (001 total): 00
  Codes of length 06 bits (001 total): 0A
  Codes of length 07 bits (001 total): 0B
  Codes of length 08 bits (000 total):
  Codes of length 09 bits (000 total):
  Codes of length 10 bits (000 total):
  Codes of length 11 bits (000 total):
  Codes of length 12 bits (000 total):
  Codes of length 13 bits (000 total):
  Codes of length 14 bits (000 total):
  Codes of length 15 bits (000 total):
  Codes of length 16 bits (000 total):
  Total number of codes: 012

----
Destination ID = 0
Class = 1 (AC Table)
  Codes of length 01 bits (000 total):
  Codes of length 02 bits (002 total): 01 02
  Codes of length 03 bits (001 total): 03
  Codes of length 04 bits (003 total): 04 11 05
  Codes of length 05 bits (004 total): 12 06 21 07
  Codes of length 06 bits (001 total): 13
  Codes of length 07 bits (003 total): 22 00 08
```

```
  Codes of length 08 bits (003 total): 31 14 41
  Codes of length 09 bits (002 total): 32 23
  Codes of length 10 bits (003 total): 15 09 51
  Codes of length 11 bits (003 total): 42 16 61
  Codes of length 12 bits (003 total): 24 33 17
  Codes of length 13 bits (002 total): 52 71
  Codes of length 14 bits (006 total): 81 18 62 91 25 43
  Codes of length 15 bits (009 total): A1 B1 F0 26 34 72 0A 19 C1
  Codes of length 16 bits (117 total):
      D1 35 27 E1 53 36 82 F1 92 A2 44 54 73 45 46 37
      47 63 28 55 56 57 1A B2 C2 D2 E2 F2 64 83 74 93
      84 65 A3 B3 C3 D3 E3 29 38 66 F3 75 2A 39 3A 48
      49 4A 58 59 5A 67 68 69 6A 76 77 78 79 7A 85 86
      87 88 89 8A 94 95 96 97 98 99 9A A4 A5 A6 A7 A8
      A9 AA B4 B5 B6 B7 B8 B9 BA C4 C5 C6 C7 C8 C9 CA
      D4 D5 D6 D7 D8 D9 DA E4 E5 E6 E7 E8 E9 EA F4 F5
      F6 F7 F8 F9 FA
  Total number of codes: 162

----
Destination ID = 1
Class = 1 (AC Table)
  Codes of length 01 bits (000 total):
  Codes of length 02 bits (002 total): 01 02
  Codes of length 03 bits (001 total): 03
  Codes of length 04 bits (003 total): 11 04 21
  Codes of length 05 bits (002 total): 12 05
  Codes of length 06 bits (004 total): 31 06 00 22
  Codes of length 07 bits (004 total): 13 41 51 07
  Codes of length 08 bits (003 total): 32 61 14
  Codes of length 09 bits (005 total): 71 08 42 81 23
  Codes of length 10 bits (004 total): 91 15 52 A1
  Codes of length 11 bits (004 total): 62 16 33 09
  Codes of length 12 bits (004 total): B1 24 C1 D1
```

```
Codes of length 13 bits (006 total): 43 72 F0 17 E1 82
Codes of length 14 bits (006 total): 34 25 92 53 18 63
Codes of length 15 bits (005 total): 44 F1 A2 B2 26
Codes of length 16 bits (109 total):
    35 19 54 36 45 64 27 0A 73 83 93 46 74 C2 D2 E2
    F2 55 65 75 56 37 84 85 A3 B3 C3 D3 E3 F3 29 1A
    94 A4 B4 C4 D4 E4 F4 95 A5 B5 C5 D5 E5 F5 28 47
    57 66 38 76 86 96 A6 B6 C6 D6 E6 F6 67 77 87 97
    A7 B7 C7 D7 E7 F7 48 58 68 78 88 98 A8 B8 C8 D8
    E8 F8 39 49 59 69 79 89 99 A9 B9 C9 D9 E9 F9 2A
    3A 4A 5A 6A 7A 8A 9A AA BA CA DA EA FA
Total number of codes: 162
```

Compression quality: 90

Luminance quantization table:

```
DQT, Row #0:   1   1   1   1   2   2   2   3
DQT, Row #1:   1   1   1   1   2   2   2   3
DQT, Row #2:   1   1   1   1   2   3   4   5
DQT, Row #3:   1   1   1   2   3   4   5   7
DQT, Row #4:   2   2   2   3   4   5   7   8
DQT, Row #5:   2   2   3   4   5   7   8   8
DQT, Row #6:   2   2   4   5   7   8   8   8
DQT, Row #7:   3   3   5   7   8   8   8   8
```

Chrominance quantization table:

```
DQT, Row #0:   1   1   2   5   7   8   8   8
DQT, Row #1:   1   2   3   5   8   8   8   8
DQT, Row #2:   2   3   4   8   8   8   8   8
DQT, Row #3:   5   5   8   8   8   8   8   8
DQT, Row #4:   7   8   8   8   8   8   8   8
DQT, Row #5:   8   8   8   8   8   8   8   8
DQT, Row #6:   8   8   8   8   8   8   8   8
DQT, Row #7:   8   8   8   8   8   8   8   8
```

Chroma subsampling not performed:

```
  Component[1]: ... (Subsamp 1 x 1), ... (Lum: Y)
  Component[2]: ... (Subsamp 1 x 1), ... (Chrom: Cb)
  Component[3]: ... (Subsamp 1 x 1), ... (Chrom: Cr)
```

Huffman tables:

```
Destination ID = 0
Class = 0 (DC / Lossless Table)
  Codes of length 01 bits (000 total):
  Codes of length 02 bits (000 total):
  Codes of length 03 bits (006 total): 07 08 06 05 04 09
  Codes of length 04 bits (002 total): 03 0A
  Codes of length 05 bits (003 total): 02 01 00
  Codes of length 06 bits (001 total): 0B
  Codes of length 07 bits (000 total):
  Codes of length 08 bits (000 total):
  Codes of length 09 bits (000 total):
  Codes of length 10 bits (000 total):
  Codes of length 11 bits (000 total):
  Codes of length 12 bits (000 total):
  Codes of length 13 bits (000 total):
  Codes of length 14 bits (000 total):
  Codes of length 15 bits (000 total):
  Codes of length 16 bits (000 total):
  Total number of codes: 012

----
Destination ID = 1
Class = 0 (DC / Lossless Table)
  Codes of length 01 bits (000 total):
  Codes of length 02 bits (000 total):
  Codes of length 03 bits (006 total): 06 05 04 03 07 02
```

```
  Codes of length 04 bits (003 total): 08 01 09
  Codes of length 05 bits (001 total): 00
  Codes of length 06 bits (001 total): 0A
  Codes of length 07 bits (001 total): 0B
  Codes of length 08 bits (000 total):
  Codes of length 09 bits (000 total):
  Codes of length 10 bits (000 total):
  Codes of length 11 bits (000 total):
  Codes of length 12 bits (000 total):
  Codes of length 13 bits (000 total):
  Codes of length 14 bits (000 total):
  Codes of length 15 bits (000 total):
  Codes of length 16 bits (000 total):
  Total number of codes: 012

----
Destination ID = 0
Class = 1 (AC Table)
  Codes of length 01 bits (000 total):
  Codes of length 02 bits (002 total): 01 02
  Codes of length 03 bits (001 total): 03
  Codes of length 04 bits (002 total): 04 11
  Codes of length 05 bits (005 total): 05 06 21 12 00
  Codes of length 06 bits (002 total): 07 31
  Codes of length 07 bits (003 total): 41 13 08
  Codes of length 08 bits (004 total): 51 22 61 14
  Codes of length 09 bits (006 total): 71 81 32 91 09 A1
  Codes of length 10 bits (006 total): 23 F0 C1 42 B1 15
  Codes of length 11 bits (005 total): D1 16 E1 F1 52
  Codes of length 12 bits (005 total): 33 17 24 62 18
  Codes of length 13 bits (001 total): 43
  Codes of length 14 bits (003 total): 34 25 82
  Codes of length 15 bits (006 total): 0A 19 72 53 26 63
  Codes of length 16 bits (111 total):
```

```
      92 44 35 A2 54 B2 1A 73 36 C2 D2 27 45 37 46 E2
      F2 83 93 A3 B3 64 55 28 C3 D3 29 38 E3 F3 47 48
      56 65 2A 39 3A 49 4A 57 58 59 5A 66 74 75 84 85
      67 76 77 68 86 87 94 95 A4 A5 B4 B5 C4 C5 D4 D5
      E4 E5 F4 F5 96 97 A6 A7 B6 B7 C6 C7 D6 D7 E6 E7
      F6 F7 69 6A 78 79 7A 88 89 8A 98 99 9A A8 A9 AA
      B8 B9 BA C8 C9 CA D8 D9 DA E8 E9 EA F8 F9 FA
  Total number of codes: 162

----
Destination ID = 1
Class = 1 (AC Table)
  Codes of length 01 bits (000 total):
  Codes of length 02 bits (001 total): 01
  Codes of length 03 bits (003 total): 02 03 11
  Codes of length 04 bits (002 total): 00 04
  Codes of length 05 bits (003 total): 21 05 12
  Codes of length 06 bits (004 total): 31 06 41 F0
  Codes of length 07 bits (007 total): 51 61 07 13 22 71 81
  Codes of length 08 bits (006 total): 91 A1 B1 C1 08 32
  Codes of length 09 bits (003 total): D1 14 E1
  Codes of length 10 bits (004 total): 23 F1 42 15
  Codes of length 11 bits (003 total): 52 09 16
  Codes of length 12 bits (006 total): 33 62 D2 72 24 82
  Codes of length 13 bits (007 total): C2 92 93 43 17 73 83
  Codes of length 14 bits (007 total): A2 B2 63 25 34 53 E2
  Codes of length 15 bits (001 total): B3
  Codes of length 16 bits (105 total):
      35 26 44 54 64 45 55 27 0A 84 B4 18 19 1A 28 29
      2A 36 37 38 39 3A 46 47 48 49 4A 56 57 58 59 5A
      65 66 67 68 69 6A 74 75 76 77 78 79 7A 85 86 87
      88 89 8A 94 95 96 97 98 99 9A A3 A4 A5 A6 A7 A8
      A9 AA B5 B6 B7 B8 B9 BA C3 C4 C5 C6 C7 C8 C9 CA
      D3 D4 D5 D6 D7 D8 D9 DA E3 E4 E5 E6 E7 E8 E9 EA
```

```
    F2 F3 F4 F5 F6 F7 F8 F9 FA
Total number of codes: 162
```

Compression quality: 80

Luminance quantization table:

```
DQT, Row #0:   2   2   2   2   3   4   5   6
DQT, Row #1:   2   2   2   2   3   4   5   6
DQT, Row #2:   2   2   2   2   4   5   7   9
DQT, Row #3:   2   2   2   4   5   7   9  12
DQT, Row #4:   3   3   4   5   8  10  12  12
DQT, Row #5:   4   4   5   7  10  12  12  12
DQT, Row #6:   5   5   7   9  12  12  12  12
DQT, Row #7:   6   6   9  12  12  12  12  12
```

Chrominance quantization table:

```
DQT, Row #0:   3   3   5   9  13  15  15  15
DQT, Row #1:   3   4   6  11  14  12  12  12
DQT, Row #2:   5   6   9  14  12  12  12  12
DQT, Row #3:   9  11  14  12  12  12  12  12
DQT, Row #4:  13  14  12  12  12  12  12  12
DQT, Row #5:  15  12  12  12  12  12  12  12
DQT, Row #6:  15  12  12  12  12  12  12  12
DQT, Row #7:  15  12  12  12  12  12  12  12
```

Chroma subsampling not performed:

```
Component[1]: ... (Subsamp 1 x 1), ... (Lum: Y)
Component[2]: ... (Subsamp 1 x 1), ... (Chrom: Cb)
Component[3]: ... (Subsamp 1 x 1), ... (Chrom: Cr)
```

Huffman tables:

```
Destination ID = 0
Class = 0 (DC / Lossless Table)
  Codes of length 01 bits (000 total):
  Codes of length 02 bits (000 total):
  Codes of length 03 bits (007 total): 04 05 03 02 06 01 00
  Codes of length 04 bits (001 total): 07
  Codes of length 05 bits (001 total): 08
  Codes of length 06 bits (001 total): 09
  Codes of length 07 bits (001 total): 0A
  Codes of length 08 bits (001 total): 0B
  Codes of length 09 bits (000 total):
  Codes of length 10 bits (000 total):
  Codes of length 11 bits (000 total):
  Codes of length 12 bits (000 total):
  Codes of length 13 bits (000 total):
  Codes of length 14 bits (000 total):
  Codes of length 15 bits (000 total):
  Codes of length 16 bits (000 total):
  Total number of codes: 012

----
Destination ID = 1
Class = 0 (DC / Lossless Table)
  Codes of length 01 bits (000 total):
  Codes of length 02 bits (002 total): 01 00
  Codes of length 03 bits (002 total): 02 03
  Codes of length 04 bits (003 total): 04 05 06
  Codes of length 05 bits (001 total): 07
  Codes of length 06 bits (001 total): 08
  Codes of length 07 bits (001 total): 09
  Codes of length 08 bits (001 total): 0A
  Codes of length 09 bits (001 total): 0B
  Codes of length 10 bits (000 total):
  Codes of length 11 bits (000 total):
```

```
  Codes of length 12 bits (000 total):
  Codes of length 13 bits (000 total):
  Codes of length 14 bits (000 total):
  Codes of length 15 bits (000 total):
  Codes of length 16 bits (000 total):
  Total number of codes: 012

----
Destination ID = 0
Class = 1 (AC Table)
  Codes of length 01 bits (000 total):
  Codes of length 02 bits (002 total): 01 02
  Codes of length 03 bits (001 total): 03
  Codes of length 04 bits (003 total): 11 04 00
  Codes of length 05 bits (003 total): 05 21 12
  Codes of length 06 bits (002 total): 31 41
  Codes of length 07 bits (004 total): 51 06 13 61
  Codes of length 08 bits (002 total): 22 71
  Codes of length 09 bits (006 total): 81 14 32 91 A1 07
  Codes of length 10 bits (007 total): 15 B1 42 23 C1 52 D1
  Codes of length 11 bits (003 total): E1 33 16
  Codes of length 12 bits (004 total): 62 F0 24 72
  Codes of length 13 bits (002 total): 82 F1
  Codes of length 14 bits (006 total): 25 43 34 53 92 A2
  Codes of length 15 bits (002 total): B2 63
  Codes of length 16 bits (115 total):
      73 C2 35 44 27 93 A3 B3 36 17 54 64 74 C3 D2 E2
      08 26 83 09 0A 18 19 84 94 45 46 A4 B4 56 D3 55
      28 1A F2 E3 F3 C4 D4 E4 F4 65 75 85 95 A5 B5 C5
      D5 E5 F5 66 76 86 96 A6 B6 C6 D6 E6 F6 37 47 57
      67 77 87 97 A7 B7 C7 D7 E7 F7 38 48 58 68 78 88
      98 A8 B8 C8 D8 E8 F8 29 39 49 59 69 79 89 99 A9
      B9 C9 D9 E9 F9 2A 3A 4A 5A 6A 7A 8A 9A AA BA CA
      DA EA FA
```

```
  Total number of codes: 162

----
Destination ID = 1
Class = 1 (AC Table)
  Codes of length 01 bits (000 total):
  Codes of length 02 bits (002 total): 01 00
  Codes of length 03 bits (002 total): 02 11
  Codes of length 04 bits (001 total): 03
  Codes of length 05 bits (002 total): 04 21
  Codes of length 06 bits (003 total): 12 31 41
  Codes of length 07 bits (005 total): 05 51 13 61 22
  Codes of length 08 bits (005 total): 06 71 81 91 32
  Codes of length 09 bits (004 total): A1 B1 F0 14
  Codes of length 10 bits (005 total): C1 D1 E1 23 42
  Codes of length 11 bits (006 total): 15 52 62 72 F1 33
  Codes of length 12 bits (004 total): 24 34 43 82
  Codes of length 13 bits (008 total): 16 92 53 25 A2 63 B2 C2
  Codes of length 14 bits (003 total): 07 73 D2
  Codes of length 15 bits (003 total): 35 E2 44
  Codes of length 16 bits (109 total):
      83 17 54 93 08 09 0A 18 19 26 36 45 1A 27 64 74
      55 37 F2 A3 B3 C3 28 29 D3 E3 F3 84 94 A4 B4 C4
      D4 E4 F4 65 75 85 95 A5 B5 C5 D5 E5 F5 46 56 66
      76 86 96 A6 B6 C6 D6 E6 F6 47 57 67 77 87 97 A7
      B7 C7 D7 E7 F7 38 48 58 68 78 88 98 A8 B8 C8 D8
      E8 F8 39 49 59 69 79 89 99 A9 B9 C9 D9 E9 F9 2A
      3A 4A 5A 6A 7A 8A 9A AA BA CA DA EA FA
  Total number of codes: 162
```

Compression level: 70

Luminance quantization table:

```
  DQT, Row #0:   4   3   3   4   6   7   8  10
```

```
DQT, Row #1:   3   3   3   4   5   6   8  10
DQT, Row #2:   3   3   3   4   6   9  12  12
DQT, Row #3:   4   4   4   7   9  12  12  17
DQT, Row #4:   6   5   6   9  12  13  17  20
DQT, Row #5:   7   6   9  12  13  17  20  20
DQT, Row #6:   8   8  12  12  17  20  20  20
DQT, Row #7:  10  10  12  17  20  20  20  20
```

Chrominance quantization table:

```
DQT, Row #0:   4   5   8  15  20  20  20  20
DQT, Row #1:   5   7  10  14  20  20  20  20
DQT, Row #2:   8  10  14  20  20  20  20  20
DQT, Row #3:  15  14  20  20  20  20  20  20
DQT, Row #4:  20  20  20  20  20  20  20  20
DQT, Row #5:  20  20  20  20  20  20  20  20
DQT, Row #6:  20  20  20  20  20  20  20  20
DQT, Row #7:  20  20  20  20  20  20  20  20
```

Chroma subsampling not performed:

```
Component[1]: ... (Subsamp 1 x 1), ... (Lum: Y)
Component[2]: ... (Subsamp 1 x 1), ... (Chrom: Cb)
Component[3]: ... (Subsamp 1 x 1), ... (Chrom: Cr)
```

Huffman tables: same as those used by compression quality 80.

Compression quality: 60

Luminance quantization table:

```
DQT, Row #0:   6   4   4   6   9  11  12  16
DQT, Row #1:   4   5   5   6   8  10  12  12
DQT, Row #2:   4   5   5   6  10  12  14  19
DQT, Row #3:   6   6   6  11  12  15  19  28
```

```
DQT, Row #4:   9   8  10  12  16  20  27  31
DQT, Row #5:  11  10  12  15  20  27  31  31
DQT, Row #6:  12  12  14  19  27  31  31  31
DQT, Row #7:  16  12  19  28  31  31  31  31
```

Chrominance quantization table:

```
DQT, Row #0:   7   7  13  24  26  31  31  31
DQT, Row #1:   7  12  16  21  31  31  31  31
DQT, Row #2:  13  16  17  31  31  31  31  31
DQT, Row #3:  24  21  31  31  31  31  31  31
DQT, Row #4:  26  31  31  31  31  31  31  31
DQT, Row #5:  31  31  31  31  31  31  31  31
DQT, Row #6:  31  31  31  31  31  31  31  31
DQT, Row #7:  31  31  31  31  31  31  31  31
```

Chroma subsampling not performed:

```
Component[1]: ... (Subsamp 1 x 1), ... (Lum: Y)
Component[2]: ... (Subsamp 1 x 1), ... (Chrom: Cb)
Component[3]: ... (Subsamp 1 x 1), ... (Chrom: Cr)
```

Huffman tables: same as those used by compression quality 80.

Compression level: 51

Luminance quantization table:

```
DQT, Row #0:   8   5   5   8  11  13  15  17
DQT, Row #1:   5   6   6   7  10  12  12  15
DQT, Row #2:   5   6   6   8  12  13  17  23
DQT, Row #3:   8   7   8  13  13  18  23  34
DQT, Row #4:  11  10  12  13  19  25  33  38
DQT, Row #5:  13  12  13  18  25  33  38  38
DQT, Row #6:  15  12  17  23  33  38  38  38
DQT, Row #7:  17  15  23  34  38  38  38  38
```

Chrominance quantization table:

```
DQT, Row #0:   8   9  16  29  32  38  38  38
DQT, Row #1:   9  14  20  26  38  38  38  38
DQT, Row #2:  16  20  21  38  38  38  38  38
DQT, Row #3:  29  26  38  38  38  38  38  38
DQT, Row #4:  32  38  38  38  38  38  38  38
DQT, Row #5:  38  38  38  38  38  38  38  38
DQT, Row #6:  38  38  38  38  38  38  38  38
DQT, Row #7:  38  38  38  38  38  38  38  38
```

Chroma subsampling not performed:

```
Component[1]: ... (Subsamp 1 x 1), ... (Lum: Y)
Component[2]: ... (Subsamp 1 x 1), ... (Chrom: Cb)
Component[3]: ... (Subsamp 1 x 1), ... (Chrom: Cr)
```

Huffman tables: same as those used by compression quality 80.

Compression quality: 50

Luminance quantization table:

```
DQT, Row #0:   8   6   6   8  12  14  16  17
DQT, Row #1:   6   6   6   8  10  13  12  15
DQT, Row #2:   6   6   7   8  13  14  18  24
DQT, Row #3:   8   8   8  14  13  19  24  35
DQT, Row #4:  12  10  13  13  20  26  34  39
DQT, Row #5:  14  13  14  19  26  34  39  39
DQT, Row #6:  16  12  18  24  34  39  39  39
DQT, Row #7:  17  15  24  35  39  39  39  39
```

Chrominance quantization table:

```
DQT, Row #0:   9   8   9  11  14  17  19  24
DQT, Row #1:   8  10   9  11  14  13  17  22
DQT, Row #2:   9   9  13  14  13  15  23  26
DQT, Row #3:  11  11  14  14  15  20  26  33
DQT, Row #4:  14  14  13  15  20  24  33  39
DQT, Row #5:  17  13  15  20  24  32  39  39
DQT, Row #6:  19  17  23  26  33  39  39  39
DQT, Row #7:  24  22  26  33  39  39  39  39
```

Chroma subsampling performed:

```
Component[1]: ... (Subsamp 1 x 1), ... (Lum: Y)
Component[2]: ... (Subsamp 2 x 2), ... (Chrom: Cb)
Component[3]: ... (Subsamp 2 x 2), ... (Chrom: Cr)
```

Huffman tables: same as those used by compression quality 80.

Compression quality: 40

Luminance quantization table:

```
DQT, Row #0:  12   8   8  12  17  21  24  23
DQT, Row #1:   8   9   9  11  15  19  18  23
DQT, Row #2:   8   9  10  12  19  20  27  36
DQT, Row #3:  12  11  12  21  20  28  36  53
DQT, Row #4:  17  15  19  20  30  39  51  59
DQT, Row #5:  21  19  20  28  39  51  59  59
DQT, Row #6:  24  18  27  36  51  59  59  59
DQT, Row #7:  23  23  36  53  59  59  59  59
```

Chrominance quantization table:

```
DQT, Row #0:  13  11  13  16  20  20  29  37
DQT, Row #1:  11  14  14  14  16  20  26  32
DQT, Row #2:  13  14  15  17  20  23  35  40
```

```
DQT, Row #3:  16  14  17  21  23  30  40  50
DQT, Row #4:  20  16  20  23  30  37  50  59
DQT, Row #5:  20  20  23  30  37  48  59  59
DQT, Row #6:  29  26  35  40  50  59  59  59
DQT, Row #7:  37  32  40  50  59  59  59  59
```

Chroma subsampling performed:

```
Component[1]: ... (Subsamp 1 x 1), ... (Lum: Y)
Component[2]: ... (Subsamp 2 x 2), ... (Chrom: Cb)
Component[3]: ... (Subsamp 2 x 2), ... (Chrom: Cr)
```

Huffman tables:

```
  Destination ID = 0
  Class = 0 (DC / Lossless Table)
    Codes of length 01 bits (000 total):
    Codes of length 02 bits (001 total): 03
    Codes of length 03 bits (005 total): 00 01 02 04 05
    Codes of length 04 bits (001 total): 06
    Codes of length 05 bits (001 total): 07
    Codes of length 06 bits (001 total): 08
    Codes of length 07 bits (001 total): 09
    Codes of length 08 bits (001 total): 0A
    Codes of length 09 bits (001 total): 0B
    Codes of length 10 bits (000 total):
    Codes of length 11 bits (000 total):
    Codes of length 12 bits (000 total):
    Codes of length 13 bits (000 total):
    Codes of length 14 bits (000 total):
    Codes of length 15 bits (000 total):
    Codes of length 16 bits (000 total):
    Total number of codes: 012
```

```
----
Destination ID = 1
Class = 0 (DC / Lossless Table)
  Codes of length 01 bits (000 total):
  Codes of length 02 bits (001 total): 01
  Codes of length 03 bits (005 total): 00 02 03 04 05
  Codes of length 04 bits (001 total): 06
  Codes of length 05 bits (001 total): 07
  Codes of length 06 bits (001 total): 08
  Codes of length 07 bits (001 total): 09
  Codes of length 08 bits (001 total): 0A
  Codes of length 09 bits (001 total): 0B
  Codes of length 10 bits (000 total):
  Codes of length 11 bits (000 total):
  Codes of length 12 bits (000 total):
  Codes of length 13 bits (000 total):
  Codes of length 14 bits (000 total):
  Codes of length 15 bits (000 total):
  Codes of length 16 bits (000 total):
  Total number of codes: 012

----
Destination ID = 0
Class = 1 (AC Table)
  Codes of length 01 bits (000 total):
  Codes of length 02 bits (001 total): 01
  Codes of length 03 bits (004 total): 00 02 11 03
  Codes of length 04 bits (001 total): 04
  Codes of length 05 bits (003 total): 21 12 31
  Codes of length 06 bits (002 total): 05 41
  Codes of length 07 bits (004 total): 51 61 13 22
  Codes of length 08 bits (002 total): 71 81
  Codes of length 09 bits (005 total): 32 06 14 91 A1
  Codes of length 10 bits (007 total): B1 42 23 24 15 52 C1
```

```
  Codes of length 11 bits (006 total): 62 33 34 72 82 D1
  Codes of length 12 bits (008 total): 43 07 25 92 53 F0 E1 F1
  Codes of length 13 bits (005 total): 63 73 35 16 A2
  Codes of length 14 bits (003 total): B2 83 26
  Codes of length 15 bits (012 total): 44 93 54 64 45 C2 A3 74 36 17 D2 55
  Codes of length 16 bits (051 total):
      E2 65 F2 B3 84 C3 D3 75 E3 F3 46 27 94 A4 85 B4
      95 C4 D4 E4 F4 A5 B5 C5 D5 E5 F5 56 66 76 86 96
      A6 B6 C6 D6 E6 F6 37 47 57 67 77 87 97 A7 B7 C7
      D7 E7 F7
  Total number of codes: 114

----
Destination ID = 1
Class = 1 (AC Table)
  Codes of length 01 bits (000 total):
  Codes of length 02 bits (002 total): 01 00
  Codes of length 03 bits (002 total): 02 11
  Codes of length 04 bits (001 total): 03
  Codes of length 05 bits (002 total): 21 31
  Codes of length 06 bits (004 total): 12 04 41 51
  Codes of length 07 bits (004 total): 61 71 22 13
  Codes of length 08 bits (003 total): 05 32 81
  Codes of length 09 bits (004 total): 91 14 A1 B1
  Codes of length 10 bits (005 total): 42 23 C1 52 D1
  Codes of length 11 bits (006 total): F0 33 24 62 E1 72
  Codes of length 12 bits (007 total): 82 92 43 53 15 63 73
  Codes of length 13 bits (007 total): 34 F1 25 06 16 A2 B2
  Codes of length 14 bits (006 total): 83 07 26 35 C2 D2
  Codes of length 15 bits (005 total): 44 93 54 A3 17
  Codes of length 16 bits (053 total):
      64 45 55 36 74 65 E2 F2 B3 84 C3 D3 75 E3 F3 46
      94 A4 85 B4 95 C4 D4 E4 F4 A5 B5 C5 D5 E5 F5 56
      66 76 86 96 A6 B6 C6 D6 E6 F6 27 37 47 57 67 77
```

```
    87 97 A7 B7 C7
Total number of codes: 111
```

Compression quality: 30

Luminance quantization table:

```
DQT, Row #0:  16  11  11  16  23  27  31  30
DQT, Row #1:  11  12  12  15  20  23  23  30
DQT, Row #2:  11  12  13  16  23  26  35  47
DQT, Row #3:  16  15  16  23  26  37  47  64
DQT, Row #4:  23  20  23  26  39  51  64  64
DQT, Row #5:  27  23  26  37  51  64  64  64
DQT, Row #6:  31  23  35  47  64  64  64  64
DQT, Row #7:  30  30  47  64  64  64  64  64
```

Chrominance quantization table:

```
DQT, Row #0:  17  15  17  21  20  26  38  48
DQT, Row #1:  15  19  18  17  20  26  35  43
DQT, Row #2:  17  18  20  22  26  30  46  53
DQT, Row #3:  21  17  22  28  30  39  53  64
DQT, Row #4:  20  20  26  30  39  48  64  64
DQT, Row #5:  26  26  30  39  48  63  64  64
DQT, Row #6:  38  35  46  53  64  64  64  64
DQT, Row #7:  48  43  53  64  64  64  64  64
```

Chroma subsampling performed:

```
Component[1]: ... (Subsamp 1 x 1), ... (Lum: Y)
Component[2]: ... (Subsamp 2 x 2), ... (Chrom: Cb)
Component[3]: ... (Subsamp 2 x 2), ... (Chrom: Cr)
```

Huffman tables: same as those used by compression quality 40.

Compression quality: 20

Luminance quantization table:

```
DQT, Row #0:  18  14  14  21  30  35  34  39
DQT, Row #1:  14  16  16  20  26  25  30  39
DQT, Row #2:  14  16  17  21  25  35  46  62
DQT, Row #3:  21  20  21  26  33  48  62  65
DQT, Row #4:  30  26  25  33  51  65  65  65
DQT, Row #5:  35  25  35  48  65  65  65  65
DQT, Row #6:  34  30  46  62  65  65  65  65
DQT, Row #7:  39  39  62  65  65  65  65  65
```

Chrominance quantization table:

```
DQT, Row #0:  20  20  22  27  26  33  49  62
DQT, Row #1:  20  25  23  22  26  33  45  56
DQT, Row #2:  22  23  26  29  33  39  59  65
DQT, Row #3:  27  22  29  36  39  51  65  65
DQT, Row #4:  26  26  33  39  51  62  65  65
DQT, Row #5:  33  33  39  51  62  65  65  65
DQT, Row #6:  49  45  59  65  65  65  65  65
DQT, Row #7:  62  56  65  65  65  65  65  65
```

Chroma subsampling performed:

```
Component[1]: ... (Subsamp 1 x 1), ... (Lum: Y)
Component[2]: ... (Subsamp 2 x 2), ... (Chrom: Cb)
Component[3]: ... (Subsamp 2 x 2), ... (Chrom: Cr)
```

Huffman tables: same as those used by compression quality 40.

Compression quality: 10

Luminance quantization table:

```
DQT, Row #0:  20  16  25  39  50  46  62  68
DQT, Row #1:  16  18  23  38  38  53  65  68
DQT, Row #2:  25  23  31  38  53  65  68  68
DQT, Row #3:  39  38  38  53  65  68  68  68
DQT, Row #4:  50  38  53  65  68  68  68  68
DQT, Row #5:  46  53  65  68  68  68  68  68
DQT, Row #6:  62  65  68  68  68  68  68  68
DQT, Row #7:  68  68  68  68  68  68  68  68
```

Chrominance quantization table:

```
DQT, Row #0:  21  25  32  38  54  68  68  68
DQT, Row #1:  25  28  24  38  54  68  68  68
DQT, Row #2:  32  24  32  43  66  68  68  68
DQT, Row #3:  38  38  43  53  68  68  68  68
DQT, Row #4:  54  54  66  68  68  68  68  68
DQT, Row #5:  68  68  68  68  68  68  68  68
DQT, Row #6:  68  68  68  68  68  68  68  68
DQT, Row #7:  68  68  68  68  68  68  68  68
```

Chroma subsampling performed:

```
Component[1]: ... (Subsamp 1 x 1), ... (Lum: Y)
Component[2]: ... (Subsamp 2 x 2), ... (Chrom: Cb)
Component[3]: ... (Subsamp 2 x 2), ... (Chrom: Cr)
```

Huffman tables:

```
  Destination ID = 0
  Class = 0 (DC / Lossless Table)
    Codes of length 01 bits (000 total):
    Codes of length 02 bits (003 total): 01 00 02
    Codes of length 03 bits (001 total): 03
    Codes of length 04 bits (001 total): 04
```

```
  Codes of length 05 bits (001 total): 05
  Codes of length 06 bits (001 total): 06
  Codes of length 07 bits (001 total): 07
  Codes of length 08 bits (001 total): 08
  Codes of length 09 bits (001 total): 09
  Codes of length 10 bits (001 total): 0A
  Codes of length 11 bits (001 total): 0B
  Codes of length 12 bits (000 total):
  Codes of length 13 bits (000 total):
  Codes of length 14 bits (000 total):
  Codes of length 15 bits (000 total):
  Codes of length 16 bits (000 total):
  Total number of codes: 012

----
Destination ID = 1
Class = 0 (DC / Lossless Table)
  Codes of length 01 bits (001 total): 00
  Codes of length 02 bits (001 total): 01
  Codes of length 03 bits (001 total): 02
  Codes of length 04 bits (001 total): 03
  Codes of length 05 bits (001 total): 04
  Codes of length 06 bits (001 total): 05
  Codes of length 07 bits (001 total): 06
  Codes of length 08 bits (001 total): 07
  Codes of length 09 bits (001 total): 08
  Codes of length 10 bits (001 total): 09
  Codes of length 11 bits (001 total): 0A
  Codes of length 12 bits (001 total): 0B
  Codes of length 13 bits (000 total):
  Codes of length 14 bits (000 total):
  Codes of length 15 bits (000 total):
  Codes of length 16 bits (000 total):
  Total number of codes: 012
```

```
----
Destination ID = 0
Class = 1 (AC Table)
  Codes of length 01 bits (000 total):
  Codes of length 02 bits (002 total): 01 00
  Codes of length 03 bits (002 total): 02 11
  Codes of length 04 bits (001 total): 03
  Codes of length 05 bits (003 total): 21 12 31
  Codes of length 06 bits (002 total): 04 41
  Codes of length 07 bits (003 total): 51 22 13
  Codes of length 08 bits (004 total): 61 71 32 81
  Codes of length 09 bits (007 total): 91 B1 42 A1 05 D1 C1
  Codes of length 10 bits (006 total): 14 F0 52 23 72 33
  Codes of length 11 bits (003 total): 62 E1 82
  Codes of length 12 bits (003 total): F1 43 34
  Codes of length 13 bits (006 total): 92 A2 B2 15 D2 53
  Codes of length 14 bits (002 total): 24 73
  Codes of length 15 bits (001 total): C2
  Codes of length 16 bits (053 total):
      63 06 83 93 E2 F2 A3 44 54 64 25 35 45 16 26 74
      36 55 65 B3 84 C3 D3 75 E3 F3 46 94 A4 85 B4 95
      C4 D4 E4 F4 A5 B5 C5 D5 E5 F5 56 66 76 86 96 A6
      B6 C6 D6 E6 F6
  Total number of codes: 098

----
Destination ID = 1
Class = 1 (AC Table)
  Codes of length 01 bits (000 total):
  Codes of length 02 bits (002 total): 00 01
  Codes of length 03 bits (002 total): 11 02
  Codes of length 04 bits (000 total):
  Codes of length 05 bits (005 total): 21 03 31 41 12
```

```
Codes of length 06 bits (001 total): 51
Codes of length 07 bits (006 total): 61 71 81 91 22 13
Codes of length 08 bits (006 total): 32 F0 A1 B1 04 C1
Codes of length 09 bits (001 total): D1
Codes of length 10 bits (003 total): E1 F1 42
Codes of length 11 bits (001 total): 52
Codes of length 12 bits (003 total): 23 62 72
Codes of length 13 bits (005 total): 14 92 33 82 43
Codes of length 14 bits (003 total): 24 A2 B2
Codes of length 15 bits (006 total): 34 53 44 63 73 C2
Codes of length 16 bits (047 total):
    D2 83 93 A3 54 E2 F2 05 15 25 06 16 26 35 64 45
    55 36 74 65 B3 84 C3 D3 75 E3 F3 46 94 A4 85 B4
    95 C4 D4 E4 F4 A5 B5 C5 D5 E5 F5 56 66 76 86
Total number of codes: 091
```

Compression quality: 0

Luminance quantization table:

```
DQT, Row #0:  27  26  41  65  66  71  71  71
DQT, Row #1:  26  29  38  47  63  71  71  71
DQT, Row #2:  41  38  47  62  71  71  71  71
DQT, Row #3:  65  47  62  71  71  71  71  71
DQT, Row #4:  66  63  71  71  71  71  71  71
DQT, Row #5:  71  71  71  71  71  71  71  71
DQT, Row #6:  71  71  71  71  71  71  71  71
DQT, Row #7:  71  71  71  71  71  71  71  71
```

Chrominance quantization table:

```
DQT, Row #0:  29  41  52  63  71  71  71  71
DQT, Row #1:  41  38  40  63  71  71  71  71
DQT, Row #2:  52  40  53  71  71  71  71  71
DQT, Row #3:  63  63  71  71  71  71  71  71
```

```
DQT, Row #4:  71  71  71  71  71  71  71  71
DQT, Row #5:  71  71  71  71  71  71  71  71
DQT, Row #6:  71  71  71  71  71  71  71  71
DQT, Row #7:  71  71  71  71  71  71  71  71
```

Chroma subsampling performed:

```
Component[1]: ... (Subsamp 1 x 1), ... (Lum: Y)
Component[2]: ... (Subsamp 2 x 2), ... (Chrom: Cb)
Component[3]: ... (Subsamp 2 x 2), ... (Chrom: Cr)
```

Huffman tables: same as those used by compression quality 10.

9 Appendix 3 - Photoshop CS5 optimized Huffman tables comparison

The AC luminance Huffman table used for `Save for web 90` is:

```
Codes of length 01 bits (000 total):
Codes of length 02 bits (002 total): 01 02
Codes of length 03 bits (001 total): 03
Codes of length 04 bits (002 total): 04 11
Codes of length 05 bits (005 total): 05 06 21 12 00
Codes of length 06 bits (002 total): 07 31
Codes of length 07 bits (003 total): 41 13 08
Codes of length 08 bits (004 total): 51 22 61 14
Codes of length 09 bits (006 total): 71 81 32 91 09 A1
Codes of length 10 bits (006 total): 23 F0 C1 42 B1 15
Codes of length 11 bits (005 total): D1 16 E1 F1 52
Codes of length 12 bits (005 total): 33 17 24 62 18
Codes of length 13 bits (001 total): 43
Codes of length 14 bits (003 total): 34 25 82
Codes of length 15 bits (006 total): 0A 19 72 53 26 63
Codes of length 16 bits (111 total):
```

```
    92 44 35 A2 54 B2 1A 73 36 C2 D2 27 45 37 46 E2
    F2 83 93 A3 B3 64 55 28 C3 D3 29 38 E3 F3 47 48
    56 65 2A 39 3A 49 4A 57 58 59 5A 66 74 75 84 85
    67 76 77 68 86 87 94 95 A4 A5 B4 B5 C4 C5 D4 D5
    E4 E5 F4 F5 96 97 A6 A7 B6 B7 C6 C7 D6 D7 E6 E7
    F6 F7 69 6A 78 79 7A 88 89 8A 98 99 9A A8 A9 AA
    B8 B9 BA C8 C9 CA D8 D9 DA E8 E9 EA F8 F9 FA
Total number of codes: 162
```

The corresponding AC luminance Huffman table used for `Save for web optimized` 90 is instead:

```
Codes of length 01 bits (001 total): 00
Codes of length 02 bits (000 total):
Codes of length 03 bits (001 total): 01
Codes of length 04 bits (003 total): 02 03 04
Codes of length 05 bits (003 total): 11 05 06
Codes of length 06 bits (001 total): 13
Codes of length 07 bits (006 total): 21 51 61 12 07 08
Codes of length 08 bits (005 total): 31 41 72 14 16
Codes of length 09 bits (003 total): 71 81 15
Codes of length 10 bits (002 total): 82 23
Codes of length 11 bits (005 total): 22 32 52 24 54
Codes of length 12 bits (003 total): 91 62 44
Codes of length 13 bits (004 total): 42 33 83 34
Codes of length 14 bits (002 total): A1 63
Codes of length 15 bits (002 total): 25 64
Codes of length 16 bits (003 total): 36 56 17
Total number of codes: 044
```

The progressive optimized `Save as` file, with compression level 12, has the following Huffman tables:

```
Destination ID = 0
Class = 0 (DC / Lossless Table)
```

```
  Codes of length 01 bits (001 total): 00
  Codes of length 02 bits (001 total): 09
  Codes of length 03 bits (001 total): 08
  Codes of length 04 bits (000 total):
  Codes of length 05 bits (002 total): 06 07
  Codes of length 06 bits (003 total): 03 04 05
  Codes of length 07 bits (001 total): 02
  Codes of length 08 bits (001 total): 01
  Codes of length 09 bits (000 total):
  Codes of length 10 bits (000 total):
  Codes of length 11 bits (000 total):
  Codes of length 12 bits (000 total):
  Codes of length 13 bits (000 total):
  Codes of length 14 bits (000 total):
  Codes of length 15 bits (000 total):
  Codes of length 16 bits (000 total):
  Total number of codes: 010

----
Destination ID = 1
Class = 0 (DC / Lossless Table)
  Codes of length 01 bits (001 total): 00
  Codes of length 02 bits (000 total):
  Codes of length 03 bits (002 total): 08 0A
  Codes of length 04 bits (002 total): 07 09
  Codes of length 05 bits (002 total): 06 0B
  Codes of length 06 bits (003 total): 03 04 05
  Codes of length 07 bits (001 total): 02
  Codes of length 08 bits (001 total): 01
  Codes of length 09 bits (000 total):
  Codes of length 10 bits (000 total):
  Codes of length 11 bits (000 total):
  Codes of length 12 bits (000 total):
  Codes of length 13 bits (000 total):
```

```
  Codes of length 14 bits (000 total):
  Codes of length 15 bits (000 total):
  Codes of length 16 bits (000 total):
  Total number of codes: 012

----
Destination ID = 0
Class = 1 (AC Table)
  Codes of length 01 bits (000 total):
  Codes of length 02 bits (000 total):
  Codes of length 03 bits (004 total): 05 06 07 08
  Codes of length 04 bits (004 total): 70 02 03 04
  Codes of length 05 bits (005 total): 60 01 15 16 09
  Codes of length 06 bits (003 total): 50 14 17
  Codes of length 07 bits (004 total): 40 80 18 0A
  Codes of length 08 bits (002 total): 30 13
  Codes of length 09 bits (002 total): 20 12
  Codes of length 10 bits (003 total): 10 11 31
  Codes of length 11 bits (001 total): 00
  Codes of length 12 bits (001 total): 21
  Codes of length 13 bits (001 total): 19
  Codes of length 14 bits (000 total):
  Codes of length 15 bits (000 total):
  Codes of length 16 bits (000 total):
  Total number of codes: 030

----
Destination ID = 1
Class = 1 (AC Table)
  Codes of length 01 bits (000 total):
  Codes of length 02 bits (001 total): 02
  Codes of length 03 bits (004 total): 01 03 04 05
  Codes of length 04 bits (000 total):
  Codes of length 05 bits (004 total): 11 21 12 06
```

```
  Codes of length 06 bits (003 total): 31 41 13
  Codes of length 07 bits (005 total): 00 70 51 14 15
  Codes of length 08 bits (004 total): 60 61 42 52
  Codes of length 09 bits (007 total): 50 71 22 32 62 73 07
  Codes of length 10 bits (007 total): 30 40 80 81 23 24 16
  Codes of length 11 bits (004 total): 91 72 82 63
  Codes of length 12 bits (002 total): 20 25
  Codes of length 13 bits (003 total): 10 53 84
  Codes of length 14 bits (001 total): 33
  Codes of length 15 bits (001 total): 92
  Codes of length 16 bits (001 total): 36
  Total number of codes: 047

----
Destination ID = 2
Class = 1 (AC Table)
  Codes of length 01 bits (000 total):
  Codes of length 02 bits (002 total): 01 02
  Codes of length 03 bits (001 total): 03
  Codes of length 04 bits (003 total): 11 04 05
  Codes of length 05 bits (001 total): 06
  Codes of length 06 bits (004 total): 21 61 12 13
  Codes of length 07 bits (007 total): 00 70 31 41 51 22 07
  Codes of length 08 bits (004 total): 60 71 32 15
  Codes of length 09 bits (005 total): 50 81 72 14 08
  Codes of length 10 bits (009 total): 40 80 91 52 62 82 23 63 73
  Codes of length 11 bits (005 total): 30 42 33 53 24
  Codes of length 12 bits (006 total): 10 20 A1 92 B3 74
  Codes of length 13 bits (005 total): 43 83 93 34 26
  Codes of length 14 bits (004 total): B2 44 25 17
  Codes of length 15 bits (001 total): A2
  Codes of length 16 bits (005 total): 64 85 16 45 36
  Total number of codes: 062
```

10 Appendix 4 - GIMP 2 "Export"

Compression quality 100

Luminance quantization table:

```
DQT, Row #0:   1   1   1   1   1   1   1   1
DQT, Row #1:   1   1   1   1   1   1   1   1
DQT, Row #2:   1   1   1   1   1   1   1   1
DQT, Row #3:   1   1   1   1   1   1   1   1
DQT, Row #4:   1   1   1   1   1   1   1   1
DQT, Row #5:   1   1   1   1   1   1   1   1
DQT, Row #6:   1   1   1   1   1   1   1   1
DQT, Row #7:   1   1   1   1   1   1   1   1
```

Chrominance quantization table:

```
DQT, Row #0:   1   1   1   1   1   1   1   1
DQT, Row #1:   1   1   1   1   1   1   1   1
DQT, Row #2:   1   1   1   1   1   1   1   1
DQT, Row #3:   1   1   1   1   1   1   1   1
DQT, Row #4:   1   1   1   1   1   1   1   1
DQT, Row #5:   1   1   1   1   1   1   1   1
DQT, Row #6:   1   1   1   1   1   1   1   1
DQT, Row #7:   1   1   1   1   1   1   1   1
```

Chroma subsampling can be selected by the user (4:4:4, 4:2:2 vertical, 4:2:2 horizontal, 4:2:0). The Huffman tables are:

```
Destination ID = 0
Class = 0 (DC / Lossless Table)
  Codes of length 01 bits (000 total):
  Codes of length 02 bits (001 total): 00
  Codes of length 03 bits (005 total): 01 02 03 04 05
```

```
  Codes of length 04 bits (001 total): 06
  Codes of length 05 bits (001 total): 07
  Codes of length 06 bits (001 total): 08
  Codes of length 07 bits (001 total): 09
  Codes of length 08 bits (001 total): 0A
  Codes of length 09 bits (001 total): 0B
  Codes of length 10 bits (000 total):
  Codes of length 11 bits (000 total):
  Codes of length 12 bits (000 total):
  Codes of length 13 bits (000 total):
  Codes of length 14 bits (000 total):
  Codes of length 15 bits (000 total):
  Codes of length 16 bits (000 total):
  Total number of codes: 012

Destination ID = 0
Class = 1 (AC Table)
  Codes of length 01 bits (000 total):
  Codes of length 02 bits (002 total): 01 02
  Codes of length 03 bits (001 total): 03
  Codes of length 04 bits (003 total): 00 04 11
  Codes of length 05 bits (003 total): 05 12 21
  Codes of length 06 bits (002 total): 31 41
  Codes of length 07 bits (004 total): 06 13 51 61
  Codes of length 08 bits (003 total): 07 22 71
  Codes of length 09 bits (005 total): 14 32 81 91 A1
  Codes of length 10 bits (005 total): 08 23 42 B1 C1
  Codes of length 11 bits (004 total): 15 52 D1 F0
  Codes of length 12 bits (004 total): 24 33 62 72
  Codes of length 13 bits (000 total):
  Codes of length 14 bits (000 total):
  Codes of length 15 bits (001 total): 82
  Codes of length 16 bits (125 total):
      09 0A 16 17 18 19 1A 25 26 27 28 29 2A 34 35 36
```

```
      37 38 39 3A 43 44 45 46 47 48 49 4A 53 54 55 56
      57 58 59 5A 63 64 65 66 67 68 69 6A 73 74 75 76
      77 78 79 7A 83 84 85 86 87 88 89 8A 92 93 94 95
      96 97 98 99 9A A2 A3 A4 A5 A6 A7 A8 A9 AA B2 B3
      B4 B5 B6 B7 B8 B9 BA C2 C3 C4 C5 C6 C7 C8 C9 CA
      D2 D3 D4 D5 D6 D7 D8 D9 DA E1 E2 E3 E4 E5 E6 E7
      E8 E9 EA F1 F2 F3 F4 F5 F6 F7 F8 F9 FA
  Total number of codes: 162

Destination ID = 1
Class = 0 (DC / Lossless Table)
  Codes of length 01 bits (000 total):
  Codes of length 02 bits (003 total): 00 01 02
  Codes of length 03 bits (001 total): 03
  Codes of length 04 bits (001 total): 04
  Codes of length 05 bits (001 total): 05
  Codes of length 06 bits (001 total): 06
  Codes of length 07 bits (001 total): 07
  Codes of length 08 bits (001 total): 08
  Codes of length 09 bits (001 total): 09
  Codes of length 10 bits (001 total): 0A
  Codes of length 11 bits (001 total): 0B
  Codes of length 12 bits (000 total):
  Codes of length 13 bits (000 total):
  Codes of length 14 bits (000 total):
  Codes of length 15 bits (000 total):
  Codes of length 16 bits (000 total):
  Total number of codes: 012

Destination ID = 1
Class = 1 (AC Table)
  Codes of length 01 bits (000 total):
  Codes of length 02 bits (002 total): 00 01
  Codes of length 03 bits (001 total): 02
```

```
Codes of length 04 bits (002 total): 03 11
Codes of length 05 bits (004 total): 04 05 21 31
Codes of length 06 bits (004 total): 06 12 41 51
Codes of length 07 bits (003 total): 07 61 71
Codes of length 08 bits (004 total): 13 22 32 81
Codes of length 09 bits (007 total): 08 14 42 91 A1 B1 C1
Codes of length 10 bits (005 total): 09 23 33 52 F0
Codes of length 11 bits (004 total): 15 62 72 D1
Codes of length 12 bits (004 total): 0A 16 24 34
Codes of length 13 bits (000 total):
Codes of length 14 bits (001 total): E1
Codes of length 15 bits (002 total): 25 F1
Codes of length 16 bits (119 total):
     17 18 19 1A 26 27 28 29 2A 35 36 37 38 39 3A 43
     44 45 46 47 48 49 4A 53 54 55 56 57 58 59 5A 63
     64 65 66 67 68 69 6A 73 74 75 76 77 78 79 7A 82
     83 84 85 86 87 88 89 8A 92 93 94 95 96 97 98 99
     9A A2 A3 A4 A5 A6 A7 A8 A9 AA B2 B3 B4 B5 B6 B7
     B8 B9 BA C2 C3 C4 C5 C6 C7 C8 C9 CA D2 D3 D4 D5
     D6 D7 D8 D9 DA E2 E3 E4 E5 E6 E7 E8 E9 EA F2 F3
     F4 F5 F6 F7 F8 F9 FA
Total number of codes: 162
```

Compression quality 99

Luminance quantization table:

```
DQT, Row #0:   1   1   1   1   1   1   1   1
DQT, Row #1:   1   1   1   1   1   1   1   1
DQT, Row #2:   1   1   1   1   1   1   1   1
DQT, Row #3:   1   1   1   1   1   2   2   1
DQT, Row #4:   1   1   1   1   1   2   2   2
DQT, Row #5:   1   1   1   1   2   2   2   2
DQT, Row #6:   1   1   2   2   2   2   2   2
DQT, Row #7:   1   2   2   2   2   2   2   2
```

Chrominance quantization table:

```
DQT, Row #0:   1   1   1   1   2   2   2   2
DQT, Row #1:   1   1   1   1   2   2   2   2
DQT, Row #2:   1   1   1   2   2   2   2   2
DQT, Row #3:   1   1   2   2   2   2   2   2
DQT, Row #4:   2   2   2   2   2   2   2   2
DQT, Row #5:   2   2   2   2   2   2   2   2
DQT, Row #6:   2   2   2   2   2   2   2   2
DQT, Row #7:   2   2   2   2   2   2   2   2
```

Chroma subsampling can be selected by the user (4:4:4, 4:2:2 vertical, 4:2:2 horizontal, 4:2:0). The Huffman tables are the same as the previous compression setting.

Compression quality 98

Luminance quantization table:

```
DQT, Row #0:   1   1   1   1   1   2   2   2
DQT, Row #1:   1   1   1   1   1   2   2   2
DQT, Row #2:   1   1   1   1   2   2   3   2
DQT, Row #3:   1   1   1   1   2   3   3   2
DQT, Row #4:   1   1   1   2   3   4   4   3
DQT, Row #5:   1   1   2   3   3   4   5   4
DQT, Row #6:   2   3   3   3   4   5   5   4
DQT, Row #7:   3   4   4   4   4   4   4   4
```

Chrominance quantization table:

```
DQT, Row #0:   1   1   1   2   4   4   4   4
DQT, Row #1:   1   1   1   3   4   4   4   4
DQT, Row #2:   1   1   2   4   4   4   4   4
DQT, Row #3:   2   3   4   4   4   4   4   4
DQT, Row #4:   4   4   4   4   4   4   4   4
```

```
DQT, Row #5:   4   4   4   4   4   4   4   4
DQT, Row #6:   4   4   4   4   4   4   4   4
DQT, Row #7:   4   4   4   4   4   4   4   4
```

Chroma subsampling can be selected by the user (4:4:4, 4:2:2 vertical, 4:2:2 horizontal, 4:2:0). The Huffman tables are the same as the previous compression setting.

Compression quality 97

Luminance quantization table:

```
DQT, Row #0:   1   1   1   1   1   2   3   4
DQT, Row #1:   1   1   1   1   2   3   4   3
DQT, Row #2:   1   1   1   1   2   3   4   3
DQT, Row #3:   1   1   1   2   3   5   5   4
DQT, Row #4:   1   1   2   3   4   7   6   5
DQT, Row #5:   1   2   3   4   5   6   7   6
DQT, Row #6:   3   4   5   5   6   7   7   6
DQT, Row #7:   4   6   6   6   7   6   6   6
```

Chrominance quantization table:

```
DQT, Row #0:   1   1   1   3   6   6   6   6
DQT, Row #1:   1   1   2   4   6   6   6   6
DQT, Row #2:   1   2   3   6   6   6   6   6
DQT, Row #3:   3   4   6   6   6   6   6   6
DQT, Row #4:   6   6   6   6   6   6   6   6
DQT, Row #5:   6   6   6   6   6   6   6   6
DQT, Row #6:   6   6   6   6   6   6   6   6
DQT, Row #7:   6   6   6   6   6   6   6   6
```

Chroma subsampling can be selected by the user (4:4:4, 4:2:2 vertical, 4:2:2 horizontal, 4:2:0). The Huffman tables are the same as the previous compression setting.

Compression quality 96

Luminance quantization table:

```
DQT, Row #0:   1   1   1   1   2   3   4   5
DQT, Row #1:   1   1   1   2   2   5   5   4
DQT, Row #2:   1   1   1   2   3   5   6   4
DQT, Row #3:   1   1   2   2   4   7   6   5
DQT, Row #4:   1   2   3   4   5   9   8   6
DQT, Row #5:   2   3   4   5   6   8   9   7
DQT, Row #6:   4   5   6   7   8  10  10   8
DQT, Row #7:   6   7   8   8   9   8   8   8
```

Chrominance quantization table:

```
DQT, Row #0:   1   1   2   4   8   8   8   8
DQT, Row #1:   1   2   2   5   8   8   8   8
DQT, Row #2:   2   2   4   8   8   8   8   8
DQT, Row #3:   4   5   8   8   8   8   8   8
DQT, Row #4:   8   8   8   8   8   8   8   8
DQT, Row #5:   8   8   8   8   8   8   8   8
DQT, Row #6:   8   8   8   8   8   8   8   8
DQT, Row #7:   8   8   8   8   8   8   8   8
```

Chroma subsampling can be selected by the user (4:4:4, 4:2:2 vertical, 4:2:2 horizontal, 4:2:0). The Huffman tables are the same as the previous compression setting.

Compression quality 95

Luminance quantization table:

```
DQT, Row #0:   2   1   1   2   2   4   5   6
DQT, Row #1:   1   1   1   2   3   6   6   6
DQT, Row #2:   1   1   2   2   4   6   7   6
DQT, Row #3:   1   2   2   3   5   9   8   6
```

```
DQT, Row #4:   2   2   4   6   7  11  10   8
DQT, Row #5:   2   4   6   6   8  10  11   9
DQT, Row #6:   5   6   8   9  10  12  12  10
DQT, Row #7:   7   9  10  10  11  10  10  10
```

Chrominance quantization table:

```
DQT, Row #0:   2   2   2   5  10  10  10  10
DQT, Row #1:   2   2   3   7  10  10  10  10
DQT, Row #2:   2   3   6  10  10  10  10  10
DQT, Row #3:   5   7  10  10  10  10  10  10
DQT, Row #4:  10  10  10  10  10  10  10  10
DQT, Row #5:  10  10  10  10  10  10  10  10
DQT, Row #6:  10  10  10  10  10  10  10  10
DQT, Row #7:  10  10  10  10  10  10  10  10
```

Chroma subsampling can be selected by the user (4:4:4, 4:2:2 vertical, 4:2:2 horizontal, 4:2:0). The Huffman tables are the same as the previous compression setting.

Compression quality 94

Luminance quantization table:

```
DQT, Row #0:   2   1   1   2   3   5   6   7
DQT, Row #1:   1   1   2   2   3   7   7   7
DQT, Row #2:   2   2   2   3   5   7   8   7
DQT, Row #3:   2   2   3   3   6  10  10   7
DQT, Row #4:   2   3   4   7   8  13  12   9
DQT, Row #5:   3   4   7   8  10  12  14  11
DQT, Row #6:   6   8   9  10  12  15  14  12
DQT, Row #7:   9  11  11  12  13  12  12  12
```

Chrominance quantization table:

```
DQT, Row #0:   2   2   3   6  12  12  12  12
DQT, Row #1:   2   3   3   8  12  12  12  12
DQT, Row #2:   3   3   7  12  12  12  12  12
DQT, Row #3:   6   8  12  12  12  12  12  12
DQT, Row #4:  12  12  12  12  12  12  12  12
DQT, Row #5:  12  12  12  12  12  12  12  12
DQT, Row #6:  12  12  12  12  12  12  12  12
DQT, Row #7:  12  12  12  12  12  12  12  12
```

Chroma subsampling can be selected by the user (4:4:4, 4:2:2 vertical, 4:2:2 horizontal, 4:2:0). The Huffman tables are the same as the previous compression setting.

Compression quality 93

Luminance quantization table:

```
DQT, Row #0:   2   2   1   2   3   6   7   9
DQT, Row #1:   2   2   2   3   4   8   8   8
DQT, Row #2:   2   2   2   3   6   8  10   8
DQT, Row #3:   2   2   3   4   7  12  11   9
DQT, Row #4:   3   3   5   8  10  15  14  11
DQT, Row #5:   3   5   8   9  11  15  16  13
DQT, Row #6:   7   9  11  12  14  17  17  14
DQT, Row #7:  10  13  13  14  16  14  14  14
```

Chrominance quantization table:

```
DQT, Row #0:   2   3   3   7  14  14  14  14
DQT, Row #1:   3   3   4   9  14  14  14  14
DQT, Row #2:   3   4   8  14  14  14  14  14
DQT, Row #3:   7   9  14  14  14  14  14  14
DQT, Row #4:  14  14  14  14  14  14  14  14
DQT, Row #5:  14  14  14  14  14  14  14  14
DQT, Row #6:  14  14  14  14  14  14  14  14
DQT, Row #7:  14  14  14  14  14  14  14  14
```

Chroma subsampling can be selected by the user (4:4:4, 4:2:2 vertical, 4:2:2 horizontal, 4:2:0). The Huffman tables are the same as the previous compression setting.

Compression quality 92

Luminance quantization table:

```
DQT, Row #0:   3   2   2   3   4   6   8  10
DQT, Row #1:   2   2   2   3   4   9  10   9
DQT, Row #2:   2   2   3   4   6   9  11   9
DQT, Row #3:   2   3   4   5   8  14  13  10
DQT, Row #4:   3   4   6   9  11  17  16  12
DQT, Row #5:   4   6   9  10  13  17  18  15
DQT, Row #6:   8  10  12  14  16  19  19  16
DQT, Row #7:  12  15  15  16  18  16  16  16
```

Chrominance quantization table:

```
DQT, Row #0:   3   3   4   8  16  16  16  16
DQT, Row #1:   3   3   4  11  16  16  16  16
DQT, Row #2:   4   4   9  16  16  16  16  16
DQT, Row #3:   8  11  16  16  16  16  16  16
DQT, Row #4:  16  16  16  16  16  16  16  16
DQT, Row #5:  16  16  16  16  16  16  16  16
DQT, Row #6:  16  16  16  16  16  16  16  16
DQT, Row #7:  16  16  16  16  16  16  16  16
```

Chroma subsampling can be selected by the user (4:4:4, 4:2:2 vertical, 4:2:2 horizontal, 4:2:0). The Huffman tables are the same as the previous compression setting.

Compression quality 91

Luminance quantization table:

```
DQT, Row #0:   3   2   2   3   4   7   9  11
DQT, Row #1:   2   2   3   3   5  10  11  10
DQT, Row #2:   3   2   3   4   7  10  12  10
DQT, Row #3:   3   3   4   5   9  16  14  11
DQT, Row #4:   3   4   7  10  12  20  19  14
DQT, Row #5:   4   6  10  12  15  19  20  17
DQT, Row #6:   9  12  14  16  19  22  22  18
DQT, Row #7:  13  17  17  18  20  18  19  18
```

Chrominance quantization table:

```
DQT, Row #0:   3   3   4   8  18  18  18  18
DQT, Row #1:   3   4   5  12  18  18  18  18
DQT, Row #2:   4   5  10  18  18  18  18  18
DQT, Row #3:   8  12  18  18  18  18  18  18
DQT, Row #4:  18  18  18  18  18  18  18  18
DQT, Row #5:  18  18  18  18  18  18  18  18
DQT, Row #6:  18  18  18  18  18  18  18  18
DQT, Row #7:  18  18  18  18  18  18  18  18
```

Chroma subsampling can be selected by the user (4:4:4, 4:2:2 vertical, 4:2:2 horizontal, 4:2:0). The Huffman tables are the same as the previous compression setting.

Compression quality 90

Luminance quantization table:

```
DQT, Row #0:   3   2   2   3   5   8  10  12
DQT, Row #1:   2   2   3   4   5  12  12  11
DQT, Row #2:   3   3   3   5   8  11  14  11
DQT, Row #3:   3   3   4   6  10  17  16  12
DQT, Row #4:   4   4   7  11  14  22  21  15
DQT, Row #5:   5   7  11  13  16  21  23  18
DQT, Row #6:  10  13  16  17  21  24  24  20
DQT, Row #7:  14  18  19  20  22  20  21  20
```

Chrominance quantization table:

```
DQT, Row #0:   3   4   5   9  20  20  20  20
DQT, Row #1:   4   4   5  13  20  20  20  20
DQT, Row #2:   5   5  11  20  20  20  20  20
DQT, Row #3:   9  13  20  20  20  20  20  20
DQT, Row #4:  20  20  20  20  20  20  20  20
DQT, Row #5:  20  20  20  20  20  20  20  20
DQT, Row #6:  20  20  20  20  20  20  20  20
DQT, Row #7:  20  20  20  20  20  20  20  20
```

Chroma subsampling can be selected by the user (4:4:4, 4:2:2 vertical, 4:2:2 horizontal, 4:2:0). The Huffman tables are the same as the previous compression setting.

Compression quality 89

Luminance quantization table:

```
DQT, Row #0:   4   2   2   4   5   9  11  13
DQT, Row #1:   3   3   3   4   6  13  13  12
DQT, Row #2:   3   3   4   5   9  13  15  12
DQT, Row #3:   3   4   5   6  11  19  18  14
DQT, Row #4:   4   5   8  12  15  24  23  17
DQT, Row #5:   5   8  12  14  18  23  25  20
DQT, Row #6:  11  14  17  19  23  27  26  22
DQT, Row #7:  16  20  21  22  25  22  23  22
```

Chrominance quantization table:

```
DQT, Row #0:   4   4   5  10  22  22  22  22
DQT, Row #1:   4   5   6  15  22  22  22  22
DQT, Row #2:   5   6  12  22  22  22  22  22
DQT, Row #3:  10  15  22  22  22  22  22  22
DQT, Row #4:  22  22  22  22  22  22  22  22
```

```
DQT, Row #5:  22  22  22  22  22  22  22  22
DQT, Row #6:  22  22  22  22  22  22  22  22
DQT, Row #7:  22  22  22  22  22  22  22  22
```

Chroma subsampling can be selected by the user (4:4:4, 4:2:2 vertical, 4:2:2 horizontal, 4:2:0). The Huffman tables are the same as the previous compression setting.

Compression quality 88

Luminance quantization table:

```
DQT, Row #0:   4   3   2   4   6  10  12  15
DQT, Row #1:   3   3   3   5   6  14  14  13
DQT, Row #2:   3   3   4   6  10  14  17  13
DQT, Row #3:   3   4   5   7  12  21  19  15
DQT, Row #4:   4   5   9  13  16  26  25  18
DQT, Row #5:   6   8  13  15  19  25  27  22
DQT, Row #6:  12  15  19  21  25  29  29  24
DQT, Row #7:  17  22  23  24  27  24  25  24
```

Chrominance quantization table:

```
DQT, Row #0:   4   4   6  11  24  24  24  24
DQT, Row #1:   4   5   6  16  24  24  24  24
DQT, Row #2:   6   6  13  24  24  24  24  24
DQT, Row #3:  11  16  24  24  24  24  24  24
DQT, Row #4:  24  24  24  24  24  24  24  24
DQT, Row #5:  24  24  24  24  24  24  24  24
DQT, Row #6:  24  24  24  24  24  24  24  24
DQT, Row #7:  24  24  24  24  24  24  24  24
```

Chroma subsampling can be selected by the user (4:4:4, 4:2:2 vertical, 4:2:2 horizontal, 4:2:0). The Huffman tables are the same as the previous compression setting.

Compression quality 87

Luminance quantization table:

```
DQT, Row #0:   4   3   3   4   6  10  13  16
DQT, Row #1:   3   3   4   5   7  15  16  14
DQT, Row #2:   4   3   4   6  10  15  18  15
DQT, Row #3:   4   4   6   8  13  23  21  16
DQT, Row #4:   5   6  10  15  18  28  27  20
DQT, Row #5:   6   9  14  17  21  27  29  24
DQT, Row #6:  13  17  20  23  27  31  31  26
DQT, Row #7:  19  24  25  25  29  26  27  26
```

Chrominance quantization table:

```
DQT, Row #0:   4   5   6  12  26  26  26  26
DQT, Row #1:   5   5   7  17  26  26  26  26
DQT, Row #2:   6   7  15  26  26  26  26  26
DQT, Row #3:  12  17  26  26  26  26  26  26
DQT, Row #4:  26  26  26  26  26  26  26  26
DQT, Row #5:  26  26  26  26  26  26  26  26
DQT, Row #6:  26  26  26  26  26  26  26  26
DQT, Row #7:  26  26  26  26  26  26  26  26
```

Chroma subsampling can be selected by the user (4:4:4, 4:2:2 vertical, 4:2:2 horizontal, 4:2:0). The Huffman tables are the same as the previous compression setting.

Compression quality 86

Luminance quantization table:

```
DQT, Row #0:   4   3   3   4   7  11  14  17
DQT, Row #1:   3   3   4   5   7  16  17  15
DQT, Row #2:   4   4   4   7  11  16  19  16
DQT, Row #3:   4   5   6   8  14  24  22  17
```

```
DQT, Row #4:   5   6  10  16  19  31  29  22
DQT, Row #5:   7  10  15  18  23  29  32  26
DQT, Row #6:  14  18  22  24  29  34  34  28
DQT, Row #7:  20  26  27  27  31  28  29  28
```

Chrominance quantization table:

```
DQT, Row #0:   5   5   7  13  28  28  28  28
DQT, Row #1:   5   6   7  18  28  28  28  28
DQT, Row #2:   7   7  16  28  28  28  28  28
DQT, Row #3:  13  18  28  28  28  28  28  28
DQT, Row #4:  28  28  28  28  28  28  28  28
DQT, Row #5:  28  28  28  28  28  28  28  28
DQT, Row #6:  28  28  28  28  28  28  28  28
DQT, Row #7:  28  28  28  28  28  28  28  28
```

Chroma subsampling can be selected by the user (4:4:4, 4:2:2 vertical, 4:2:2 horizontal, 4:2:0). The Huffman tables are the same as the previous compression setting.

Compression quality 85

Luminance quantization table:

```
DQT, Row #0:   5   3   3   5   7  12  15  18
DQT, Row #1:   4   4   4   6   8  17  18  17
DQT, Row #2:   4   4   5   7  12  17  21  17
DQT, Row #3:   4   5   7   9  15  26  24  19
DQT, Row #4:   5   7  11  17  20  33  31  23
DQT, Row #5:   7  11  17  19  24  31  34  28
DQT, Row #6:  15  19  23  26  31  36  36  30
DQT, Row #7:  22  28  29  29  34  30  31  30
```

Chrominance quantization table:

```
DQT, Row #0:   5   5   7  14  30  30  30  30
DQT, Row #1:   5   6   8  20  30  30  30  30
DQT, Row #2:   7   8  17  30  30  30  30  30
DQT, Row #3:  14  20  30  30  30  30  30  30
DQT, Row #4:  30  30  30  30  30  30  30  30
DQT, Row #5:  30  30  30  30  30  30  30  30
DQT, Row #6:  30  30  30  30  30  30  30  30
DQT, Row #7:  30  30  30  30  30  30  30  30
```

Chroma subsampling can be selected by the user (4:4:4, 4:2:2 vertical, 4:2:2 horizontal, 4:2:0). The Huffman tables are the same as the previous compression setting.

Compression quality 84

Luminance quantization table:

```
DQT, Row #0:   5   4   3   5   8  13  16  20
DQT, Row #1:   4   4   4   6   8  19  19  18
DQT, Row #2:   4   4   5   8  13  18  22  18
DQT, Row #3:   4   5   7   9  16  28  26  20
DQT, Row #4:   6   7  12  18  22  35  33  25
DQT, Row #5:   8  11  18  20  26  33  36  29
DQT, Row #6:  16  20  25  28  33  39  38  32
DQT, Row #7:  23  29  30  31  36  32  33  32
```

Chrominance quantization table:

```
DQT, Row #0:   5   6   8  15  32  32  32  32
DQT, Row #1:   6   7   8  21  32  32  32  32
DQT, Row #2:   8   8  18  32  32  32  32  32
DQT, Row #3:  15  21  32  32  32  32  32  32
DQT, Row #4:  32  32  32  32  32  32  32  32
DQT, Row #5:  32  32  32  32  32  32  32  32
DQT, Row #6:  32  32  32  32  32  32  32  32
DQT, Row #7:  32  32  32  32  32  32  32  32
```

Chroma subsampling can be selected by the user (4:4:4, 4:2:2 vertical, 4:2:2 horizontal, 4:2:0). The Huffman tables are the same as the previous compression setting.

Compression quality 83

Luminance quantization table:

```
DQT, Row #0:   5   4   3   5   8  14  17  21
DQT, Row #1:   4   4   5   6   9  20  20  19
DQT, Row #2:   5   4   5   8  14  19  23  19
DQT, Row #3:   5   6   7  10  17  30  27  21
DQT, Row #4:   6   7  13  19  23  37  35  26
DQT, Row #5:   8  12  19  22  28  35  38  31
DQT, Row #6:  17  22  27  30  35  41  41  34
DQT, Row #7:  24  31  32  33  38  34  35  34
```

Chrominance quantization table:

```
DQT, Row #0:   6   6   8  16  34  34  34  34
DQT, Row #1:   6   7   9  22  34  34  34  34
DQT, Row #2:   8   9  19  34  34  34  34  34
DQT, Row #3:  16  22  34  34  34  34  34  34
DQT, Row #4:  34  34  34  34  34  34  34  34
DQT, Row #5:  34  34  34  34  34  34  34  34
DQT, Row #6:  34  34  34  34  34  34  34  34
DQT, Row #7:  34  34  34  34  34  34  34  34
```

Chroma subsampling can be selected by the user (4:4:4, 4:2:2 vertical, 4:2:2 horizontal, 4:2:0). The Huffman tables are the same as the previous compression setting.

Compression quality 82

Luminance quantization table:

```
DQT, Row #0:   6   4   4   6   9  14  18  22
DQT, Row #1:   4   4   5   7   9  21  22  20
DQT, Row #2:   5   5   6   9  14  21  25  20
DQT, Row #3:   5   6   8  10  18  31  29  22
DQT, Row #4:   6   8  13  20  24  39  37  28
DQT, Row #5:   9  13  20  23  29  37  41  33
DQT, Row #6:  18  23  28  31  37  44  43  36
DQT, Row #7:  26  33  34  35  40  36  37  36
```

Chrominance quantization table:

```
DQT, Row #0:   6   6   9  17  36  36  36  36
DQT, Row #1:   6   8   9  24  36  36  36  36
DQT, Row #2:   9   9  20  36  36  36  36  36
DQT, Row #3:  17  24  36  36  36  36  36  36
DQT, Row #4:  36  36  36  36  36  36  36  36
DQT, Row #5:  36  36  36  36  36  36  36  36
DQT, Row #6:  36  36  36  36  36  36  36  36
DQT, Row #7:  36  36  36  36  36  36  36  36
```

Chroma subsampling can be selected by the user (4:4:4, 4:2:2 vertical, 4:2:2 horizontal, 4:2:0). The Huffman tables are the same as the previous compression setting.

Compression quality 81

Luminance quantization table:

```
DQT, Row #0:   6   4   4   6   9  15  19  23
DQT, Row #1:   5   5   5   7  10  22  23  21
DQT, Row #2:   5   5   6   9  15  22  26  21
DQT, Row #3:   5   6   8  11  19  33  30  24
DQT, Row #4:   7   8  14  21  26  41  39  29
DQT, Row #5:   9  13  21  24  31  40  43  35
DQT, Row #6:  19  24  30  33  39  46  46  38
DQT, Row #7:  27  35  36  37  43  38  39  38
```

Chrominance quantization table:

```
DQT, Row #0:   6   7   9  18  38  38  38  38
DQT, Row #1:   7   8  10  25  38  38  38  38
DQT, Row #2:   9  10  21  38  38  38  38  38
DQT, Row #3:  18  25  38  38  38  38  38  38
DQT, Row #4:  38  38  38  38  38  38  38  38
DQT, Row #5:  38  38  38  38  38  38  38  38
DQT, Row #6:  38  38  38  38  38  38  38  38
DQT, Row #7:  38  38  38  38  38  38  38  38
```

Chroma subsampling can be selected by the user (4:4:4, 4:2:2 vertical, 4:2:2 horizontal, 4:2:0). The Huffman tables are the same as the previous compression setting.

Compression quality 80

Luminance quantization table:

```
DQT, Row #0:   6   4   4   6  10  16  20  24
DQT, Row #1:   5   5   6   8  10  23  24  22
DQT, Row #2:   6   5   6  10  16  23  28  22
DQT, Row #3:   6   7   9  12  20  35  32  25
DQT, Row #4:   7   9  15  22  27  44  41  31
DQT, Row #5:  10  14  22  26  32  42  45  37
DQT, Row #6:  20  26  31  35  41  48  48  40
DQT, Row #7:  29  37  38  39  45  40  41  40
```

Chrominance quantization table:

```
DQT, Row #0:   7   7  10  19  40  40  40  40
DQT, Row #1:   7   8  10  26  40  40  40  40
DQT, Row #2:  10  10  22  40  40  40  40  40
DQT, Row #3:  19  26  40  40  40  40  40  40
DQT, Row #4:  40  40  40  40  40  40  40  40
```

```
DQT, Row #5:  40  40  40  40  40  40  40  40
DQT, Row #6:  40  40  40  40  40  40  40  40
DQT, Row #7:  40  40  40  40  40  40  40  40
```

Chroma subsampling can be selected by the user (4:4:4, 4:2:2 vertical, 4:2:2 horizontal, 4:2:0). The Huffman tables are the same as the previous compression setting.

Compression quality 70

Luminance quantization table:

```
DQT, Row #0:  10   7   6  10  14  24  31  37
DQT, Row #1:   7   7   8  11  16  35  36  33
DQT, Row #2:   8   8  10  14  24  34  41  34
DQT, Row #3:   8  10  13  17  31  52  48  37
DQT, Row #4:  11  13  22  34  41  65  62  46
DQT, Row #5:  14  21  33  38  49  62  68  55
DQT, Row #6:  29  38  47  52  62  73  72  61
DQT, Row #7:  43  55  57  59  67  60  62  59
```

Chrominance quantization table:

```
DQT, Row #0:  10  11  14  28  59  59  59  59
DQT, Row #1:  11  13  16  40  59  59  59  59
DQT, Row #2:  14  16  34  59  59  59  59  59
DQT, Row #3:  28  40  59  59  59  59  59  59
DQT, Row #4:  59  59  59  59  59  59  59  59
DQT, Row #5:  59  59  59  59  59  59  59  59
DQT, Row #6:  59  59  59  59  59  59  59  59
DQT, Row #7:  59  59  59  59  59  59  59  59
```

Chroma subsampling can be selected by the user (4:4:4, 4:2:2 vertical, 4:2:2 horizontal, 4:2:0). The Huffman tables are the same as the previous compression setting.

Compression quality 60

Luminance quantization table:

```
DQT, Row #0:  13   9   8  13  19  32  41  49
DQT, Row #1:  10  10  11  15  21  46  48  44
DQT, Row #2:  11  10  13  19  32  46  55  45
DQT, Row #3:  11  14  18  23  41  70  64  50
DQT, Row #4:  14  18  30  45  54  87  82  62
DQT, Row #5:  19  28  44  51  65  83  90  74
DQT, Row #6:  39  51  62  70  82  97  96  81
DQT, Row #7:  58  74  76  78  90  80  82  79
```

Chrominance quantization table:

```
DQT, Row #0:  14  14  19  38  79  79  79  79
DQT, Row #1:  14  17  21  53  79  79  79  79
DQT, Row #2:  19  21  45  79  79  79  79  79
DQT, Row #3:  38  53  79  79  79  79  79  79
DQT, Row #4:  79  79  79  79  79  79  79  79
DQT, Row #5:  79  79  79  79  79  79  79  79
DQT, Row #6:  79  79  79  79  79  79  79  79
DQT, Row #7:  79  79  79  79  79  79  79  79
```

Chroma subsampling can be selected by the user (4:4:4, 4:2:2 vertical, 4:2:2 horizontal, 4:2:0). The Huffman tables are the same as the previous compression setting.

Compression quality 50

Luminance quantization table:

```
DQT, Row #0:  16  11  10  16  24  40  51  61
DQT, Row #1:  12  12  14  19  26  58  60  55
DQT, Row #2:  14  13  16  24  40  57  69  56
DQT, Row #3:  14  17  22  29  51  87  80  62
```

```
DQT, Row #4:  18  22  37  56  68 109 103  77
DQT, Row #5:  24  35  55  64  81 104 113  92
DQT, Row #6:  49  64  78  87 103 121 120 101
DQT, Row #7:  72  92  95  98 112 100 103  99
```

Chrominance quantization table:

```
DQT, Row #0:  17  18  24  47  99  99  99  99
DQT, Row #1:  18  21  26  66  99  99  99  99
DQT, Row #2:  24  26  56  99  99  99  99  99
DQT, Row #3:  47  66  99  99  99  99  99  99
DQT, Row #4:  99  99  99  99  99  99  99  99
DQT, Row #5:  99  99  99  99  99  99  99  99
DQT, Row #6:  99  99  99  99  99  99  99  99
DQT, Row #7:  99  99  99  99  99  99  99  99
```

Chroma subsampling can be selected by the user (4:4:4, 4:2:2 vertical, 4:2:2 horizontal, 4:2:0). The Huffman tables are the same as the previous compression setting.

Compression quality 40

Luminance quantization table:

```
DQT, Row #0:  20  14  13  20  30  50  64  76
DQT, Row #1:  15  15  18  24  33  73  75  69
DQT, Row #2:  18  16  20  30  50  71  86  70
DQT, Row #3:  18  21  28  36  64 109 100  78
DQT, Row #4:  23  28  46  70  85 136 129  96
DQT, Row #5:  30  44  69  80 101 130 141 115
DQT, Row #6:  61  80  98 109 129 151 150 126
DQT, Row #7:  90 115 119 123 140 125 129 124
```

Chrominance quantization table:

```
DQT, Row #0:  21  23  30  59 124 124 124 124
DQT, Row #1:  23  26  33  83 124 124 124 124
DQT, Row #2:  30  33  70 124 124 124 124 124
DQT, Row #3:  59  83 124 124 124 124 124 124
DQT, Row #4: 124 124 124 124 124 124 124 124
DQT, Row #5: 124 124 124 124 124 124 124 124
DQT, Row #6: 124 124 124 124 124 124 124 124
DQT, Row #7: 124 124 124 124 124 124 124 124
```

Chroma subsampling can be selected by the user (4:4:4, 4:2:2 vertical, 4:2:2 horizontal, 4:2:0). The Huffman tables are the same as the previous compression setting.

Compression quality 30

Luminance quantization table:

```
DQT, Row #0:  27  18  17  27  40  66  85 101
DQT, Row #1:  20  20  23  32  43  96 100  91
DQT, Row #2:  23  22  27  40  66  95 115  93
DQT, Row #3:  23  28  37  48  85 144 133 103
DQT, Row #4:  30  37  61  93 113 181 171 128
DQT, Row #5:  40  58  91 106 134 173 188 153
DQT, Row #6:  81 106 129 144 171 201 199 168
DQT, Row #7: 120 153 158 163 186 166 171 164
```

Chrominance quantization table:

```
DQT, Row #0:  28  30  40  78 164 164 164 164
DQT, Row #1:  30  35  43 110 164 164 164 164
DQT, Row #2:  40  43  93 164 164 164 164 164
DQT, Row #3:  78 110 164 164 164 164 164 164
DQT, Row #4: 164 164 164 164 164 164 164 164
DQT, Row #5: 164 164 164 164 164 164 164 164
DQT, Row #6: 164 164 164 164 164 164 164 164
DQT, Row #7: 164 164 164 164 164 164 164 164
```

Chroma subsampling can be selected by the user (4:4:4, 4:2:2 vertical, 4:2:2 horizontal, 4:2:0). The Huffman tables are the same as the previous compression setting.

Compression quality 20

Luminance quantization table:

```
DQT, Row #0:  40  28  25  40  60 100 128 153
DQT, Row #1:  30  30  35  48  65 145 150 138
DQT, Row #2:  35  33  40  60 100 143 173 140
DQT, Row #3:  35  43  55  73 128 218 200 155
DQT, Row #4:  45  55  93 140 170 255 255 193
DQT, Row #5:  60  88 138 160 203 255 255 230
DQT, Row #6: 123 160 195 218 255 255 255 253
DQT, Row #7: 180 230 238 245 255 250 255 248
```

Chrominance quantization table:

```
DQT, Row #0:  43  45  60 118 248 248 248 248
DQT, Row #1:  45  53  65 165 248 248 248 248
DQT, Row #2:  60  65 140 248 248 248 248 248
DQT, Row #3: 118 165 248 248 248 248 248 248
DQT, Row #4: 248 248 248 248 248 248 248 248
DQT, Row #5: 248 248 248 248 248 248 248 248
DQT, Row #6: 248 248 248 248 248 248 248 248
DQT, Row #7: 248 248 248 248 248 248 248 248
```

Chroma subsampling can be selected by the user (4:4:4, 4:2:2 vertical, 4:2:2 horizontal, 4:2:0). The Huffman tables are the same as the previous compression setting.

Compression quality 10

Luminance quantization table:

```
DQT, Row #0:  80  55  50  80 120 200 255 255
DQT, Row #1:  60  60  70  95 130 255 255 255
DQT, Row #2:  70  65  80 120 200 255 255 255
DQT, Row #3:  70  85 110 145 255 255 255 255
DQT, Row #4:  90 110 185 255 255 255 255 255
DQT, Row #5: 120 175 255 255 255 255 255 255
DQT, Row #6: 245 255 255 255 255 255 255 255
DQT, Row #7: 255 255 255 255 255 255 255 255
```

Chrominance quantization table:

```
DQT, Row #0:  85  90 120 235 255 255 255 255
DQT, Row #1:  90 105 130 255 255 255 255 255
DQT, Row #2: 120 130 255 255 255 255 255 255
DQT, Row #3: 235 255 255 255 255 255 255 255
DQT, Row #4: 255 255 255 255 255 255 255 255
DQT, Row #5: 255 255 255 255 255 255 255 255
DQT, Row #6: 255 255 255 255 255 255 255 255
DQT, Row #7: 255 255 255 255 255 255 255 255
```

Chroma subsampling can be selected by the user (4:4:4, 4:2:2 vertical, 4:2:2 horizontal, 4:2:0). The Huffman tables are the same as the previous compression setting.

Compression quality 0

Luminance quantization table:

```
DQT, Row #0: 255 255 255 255 255 255 255 255
DQT, Row #1: 255 255 255 255 255 255 255 255
DQT, Row #2: 255 255 255 255 255 255 255 255
DQT, Row #3: 255 255 255 255 255 255 255 255
DQT, Row #4: 255 255 255 255 255 255 255 255
DQT, Row #5: 255 255 255 255 255 255 255 255
DQT, Row #6: 255 255 255 255 255 255 255 255
DQT, Row #7: 255 255 255 255 255 255 255 255
```

Chrominance quantization table:

```
DQT, Row #0: 255 255 255 255 255 255 255 255
DQT, Row #1: 255 255 255 255 255 255 255 255
DQT, Row #2: 255 255 255 255 255 255 255 255
DQT, Row #3: 255 255 255 255 255 255 255 255
DQT, Row #4: 255 255 255 255 255 255 255 255
DQT, Row #5: 255 255 255 255 255 255 255 255
DQT, Row #6: 255 255 255 255 255 255 255 255
DQT, Row #7: 255 255 255 255 255 255 255 255
```

Chroma subsampling can be selected by the user (4:4:4, 4:2:2 vertical, 4:2:2 horizontal, 4:2:0). The Huffman tables are the same as the previous compression setting.

11 Appendix 5 - Facebook compression

Standard quality

Luminance quantization table:

```
DQT, Row #0:   9   6   6   9  14  23  30  35
DQT, Row #1:   7   7   8  11  15  34  35  32
DQT, Row #2:   8   8   9  14  23  33  40  32
DQT, Row #3:   8  10  13  17  30  50  46  36
DQT, Row #4:  10  13  21  32  39  63  60  45
DQT, Row #5:  14  20  32  37  47  60  66  53
DQT, Row #6:  28  37  45  50  60  70  70  59
DQT, Row #7:  42  53  55  57  65  58  60  57
```

Chrominance quantization table:

```
DQT, Row #0:  10  10  14  27  57  57  57  57
DQT, Row #1:  10  12  15  38  57  57  57  57
DQT, Row #2:  14  15  32  57  57  57  57  57
```

```
    DQT, Row #3:  27  38  57  57  57  57  57  57
    DQT, Row #4:  57  57  57  57  57  57  57  57
    DQT, Row #5:  57  57  57  57  57  57  57  57
    DQT, Row #6:  57  57  57  57  57  57  57  57
    DQT, Row #7:  57  57  57  57  57  57  57  57
```

Chroma subsampling:

```
    Component[1]: ... (Subsamp 1 x 1), ... (Lum: Y)
    Component[2]: ... (Subsamp 2 x 2), ... (Chrom: Cb)
    Component[3]: ... (Subsamp 2 x 2), ... (Chrom: Cr)
```

Huffman tables with scan in which they are used:

```
*** Marker: DHT (Define Huffman Table) (xFFC4) ***
  OFFSET: 0x00000D49
  Huffman table length = 26
  ----
  Destination ID = 0
  Class = 0 (DC / Lossless Table)
    Codes of length 01 bits (001 total): 00
    Codes of length 02 bits (001 total): 05
    Codes of length 03 bits (001 total): 04
    Codes of length 04 bits (001 total): 01
    Codes of length 05 bits (001 total): 03
    Codes of length 06 bits (001 total): 02
    Codes of length 07 bits (001 total): 06
    Codes of length 08 bits (000 total):
    Codes of length 09 bits (000 total):
    Codes of length 10 bits (000 total):
    Codes of length 11 bits (000 total):
    Codes of length 12 bits (000 total):
    Codes of length 13 bits (000 total):
    Codes of length 14 bits (000 total):
```

```
    Codes of length 15 bits (000 total):
    Codes of length 16 bits (000 total):
    Total number of codes: 007


*** Marker: DHT (Define Huffman Table) (xFFC4) ***
  OFFSET: 0x00000D65
  Huffman table length = 26
  ----
  Destination ID = 1
  Class = 0 (DC / Lossless Table)
    Codes of length 01 bits (001 total): 00
    Codes of length 02 bits (000 total):
    Codes of length 03 bits (002 total): 04 06
    Codes of length 04 bits (003 total): 02 03 05
    Codes of length 05 bits (001 total): 01
    Codes of length 06 bits (000 total):
    Codes of length 07 bits (000 total):
    Codes of length 08 bits (000 total):
    Codes of length 09 bits (000 total):
    Codes of length 10 bits (000 total):
    Codes of length 11 bits (000 total):
    Codes of length 12 bits (000 total):
    Codes of length 13 bits (000 total):
    Codes of length 14 bits (000 total):
    Codes of length 15 bits (000 total):
    Codes of length 16 bits (000 total):
    Total number of codes: 007


*** Marker: DHT (Define Huffman Table) (xFFC4) ***
  OFFSET: 0x00000D81
  Huffman table length = 26
  ----
```

```
  Destination ID = 1
  Class = 0 (DC / Lossless Table)
    Codes of length 01 bits (001 total): 00
    Codes of length 02 bits (000 total):
    Codes of length 03 bits (002 total): 04 06
    Codes of length 04 bits (003 total): 02 03 05
    Codes of length 05 bits (001 total): 01
    Codes of length 06 bits (000 total):
    Codes of length 07 bits (000 total):
    Codes of length 08 bits (000 total):
    Codes of length 09 bits (000 total):
    Codes of length 10 bits (000 total):
    Codes of length 11 bits (000 total):
    Codes of length 12 bits (000 total):
    Codes of length 13 bits (000 total):
    Codes of length 14 bits (000 total):
    Codes of length 15 bits (000 total):
    Codes of length 16 bits (000 total):
    Total number of codes: 007


*** Marker: SOS (Start of Scan) (xFFDA) ***
  OFFSET: 0x00000D9D
  Scan header length = 12
  Number of img components = 3
    Component[1]: selector=0x00, table=0(DC),0(AC)
    Component[2]: selector=0x01, table=1(DC),1(AC)
    Component[3]: selector=0x02, table=1(DC),1(AC)
  Spectral selection = 0 .. 0
  Successive approximation = 0x01

  NOTE: Scan parsing doesn't support this SOF mode.

*** Marker: DHT (Define Huffman Table) (xFFC4) ***
```

```
  OFFSET: 0x00003962
  Huffman table length = 35
  ----
  Destination ID = 0
  Class = 1 (AC Table)
    Codes of length 01 bits (000 total):
    Codes of length 02 bits (001 total): 02
    Codes of length 03 bits (003 total): 01 03 60
    Codes of length 04 bits (004 total): 04 11 12 50
    Codes of length 05 bits (002 total): 21 40
    Codes of length 06 bits (003 total): 20 22 30
    Codes of length 07 bits (001 total): 00
    Codes of length 08 bits (001 total): 10
    Codes of length 09 bits (001 total): 31
    Codes of length 10 bits (000 total):
    Codes of length 11 bits (000 total):
    Codes of length 12 bits (000 total):
    Codes of length 13 bits (000 total):
    Codes of length 14 bits (000 total):
    Codes of length 15 bits (000 total):
    Codes of length 16 bits (000 total):
    Total number of codes: 016


*** Marker: SOS (Start of Scan) (xFFDA) ***
  OFFSET: 0x00003987
  Scan header length = 8
  Number of img components = 1
    Component[1]: selector=0x00, table=0(DC),0(AC)
  Spectral selection = 1 .. 5
  Successive approximation = 0x02

  NOTE: Scan parsing doesn't support this SOF mode.
```

```
*** Marker: DHT (Define Huffman Table) (xFFC4) ***
  OFFSET: 0x000043E3
  Huffman table length = 32
  ----
  Destination ID = 1
  Class = 1 (AC Table)
    Codes of length 01 bits (000 total):
    Codes of length 02 bits (002 total): 01 11
    Codes of length 03 bits (002 total): 03 50
    Codes of length 04 bits (002 total): 02 40
    Codes of length 05 bits (003 total): 21 30 41
    Codes of length 06 bits (001 total): 20
    Codes of length 07 bits (001 total): 00
    Codes of length 08 bits (001 total): 10
    Codes of length 09 bits (001 total): 70
    Codes of length 10 bits (000 total):
    Codes of length 11 bits (000 total):
    Codes of length 12 bits (000 total):
    Codes of length 13 bits (000 total):
    Codes of length 14 bits (000 total):
    Codes of length 15 bits (000 total):
    Codes of length 16 bits (000 total):
    Total number of codes: 013


*** Marker: SOS (Start of Scan) (xFFDA) ***
  OFFSET: 0x00004405
  Scan header length = 8
  Number of img components = 1
    Component[1]: selector=0x02, table=1(DC),1(AC)
  Spectral selection = 1 .. 63
  Successive approximation = 0x01

  NOTE: Scan parsing doesn't support this SOF mode.
```

```
*** Marker: DHT (Define Huffman Table) (xFFC4) ***
  OFFSET: 0x00004850
  Huffman table length = 42
  ----
  Destination ID = 1
  Class = 1 (AC Table)
    Codes of length 01 bits (000 total):
    Codes of length 02 bits (001 total): 03
    Codes of length 03 bits (002 total): 01 02
    Codes of length 04 bits (003 total): 05 13 50
    Codes of length 05 bits (008 total): 00 04 11 12 21 40 41 51
    Codes of length 06 bits (002 total): 30 61
    Codes of length 07 bits (003 total): 10 20 71
    Codes of length 08 bits (001 total): 32
    Codes of length 09 bits (001 total): 81
    Codes of length 10 bits (001 total): 31
    Codes of length 11 bits (001 total): 60
    Codes of length 12 bits (000 total):
    Codes of length 13 bits (000 total):
    Codes of length 14 bits (000 total):
    Codes of length 15 bits (000 total):
    Codes of length 16 bits (000 total):
    Total number of codes: 023


*** Marker: SOS (Start of Scan) (xFFDA) ***
  OFFSET: 0x0000487C
  Scan header length = 8
  Number of img components = 1
    Component[1]: selector=0x01, table=1(DC),1(AC)
  Spectral selection = 1 .. 63
  Successive approximation = 0x01
```

```
  NOTE: Scan parsing doesn't support this SOF mode.

*** Marker: DHT (Define Huffman Table) (xFFC4) ***
  OFFSET: 0x000052B7
  Huffman table length = 34
  ----
  Destination ID = 0
  Class = 1 (AC Table)
    Codes of length 01 bits (000 total):
    Codes of length 02 bits (002 total): 01 60
    Codes of length 03 bits (001 total): 11
    Codes of length 04 bits (004 total): 02 40 50 51
    Codes of length 05 bits (003 total): 00 30 71
    Codes of length 06 bits (001 total): 20
    Codes of length 07 bits (001 total): 61
    Codes of length 08 bits (001 total): 10
    Codes of length 09 bits (001 total): 12
    Codes of length 10 bits (001 total): 80
    Codes of length 11 bits (000 total):
    Codes of length 12 bits (000 total):
    Codes of length 13 bits (000 total):
    Codes of length 14 bits (000 total):
    Codes of length 15 bits (000 total):
    Codes of length 16 bits (000 total):
    Total number of codes: 015


*** Marker: SOS (Start of Scan) (xFFDA) ***
  OFFSET: 0x000052DB
  Scan header length = 8
  Number of img components = 1
    Component[1]: selector=0x00, table=0(DC),0(AC)
  Spectral selection = 6 .. 63
  Successive approximation = 0x02
```

```
  NOTE: Scan parsing doesn't support this SOF mode.

*** Marker: DHT (Define Huffman Table) (xFFC4) ***
  OFFSET: 0x0000586A
  Huffman table length = 38
  ----
  Destination ID = 0
  Class = 1 (AC Table)
    Codes of length 01 bits (001 total): 01
    Codes of length 02 bits (000 total):
    Codes of length 03 bits (000 total):
    Codes of length 04 bits (005 total): 11 21 41 60 61
    Codes of length 05 bits (004 total): 00 31 50 51
    Codes of length 06 bits (002 total): 30 40
    Codes of length 07 bits (002 total): 20 71
    Codes of length 08 bits (003 total): 81 A1 B1
    Codes of length 09 bits (001 total): 10
    Codes of length 10 bits (001 total): 70
    Codes of length 11 bits (000 total):
    Codes of length 12 bits (000 total):
    Codes of length 13 bits (000 total):
    Codes of length 14 bits (000 total):
    Codes of length 15 bits (000 total):
    Codes of length 16 bits (000 total):
    Total number of codes: 019


*** Marker: SOS (Start of Scan) (xFFDA) ***
  OFFSET: 0x00005892
  Scan header length = 8
  Number of img components = 1
    Component[1]: selector=0x00, table=0(DC),0(AC)
  Spectral selection = 1 .. 63
```

```
  Successive approximation = 0x21

  NOTE: Scan parsing doesn't support this SOF mode.

*** Marker: SOS (Start of Scan) (xFFDA) ***
  OFFSET: 0x000063AE
  Scan header length = 12
  Number of img components = 3
    Component[1]: selector=0x00, table=0(DC),0(AC)
    Component[2]: selector=0x01, table=1(DC),1(AC)
    Component[3]: selector=0x02, table=1(DC),1(AC)
  Spectral selection = 0 .. 0
  Successive approximation = 0x10

  NOTE: Scan parsing doesn't support this SOF mode.

*** Marker: DHT (Define Huffman Table) (xFFC4) ***
  OFFSET: 0x000085DC
  Huffman table length = 34
  ----
  Destination ID = 1
  Class = 1 (AC Table)
    Codes of length 01 bits (000 total):
    Codes of length 02 bits (002 total): 01 11
    Codes of length 03 bits (001 total): 50
    Codes of length 04 bits (004 total): 00 21 40 51
    Codes of length 05 bits (002 total): 30 61
    Codes of length 06 bits (003 total): 10 20 31
    Codes of length 07 bits (001 total): 81
    Codes of length 08 bits (001 total): 71
    Codes of length 09 bits (001 total): 70
    Codes of length 10 bits (000 total):
    Codes of length 11 bits (000 total):
    Codes of length 12 bits (000 total):
```

```
    Codes of length 13 bits (000 total):
    Codes of length 14 bits (000 total):
    Codes of length 15 bits (000 total):
    Codes of length 16 bits (000 total):
    Total number of codes: 015


*** Marker: SOS (Start of Scan) (xFFDA) ***
  OFFSET: 0x00008600
  Scan header length = 8
  Number of img components = 1
    Component[1]: selector=0x02, table=1(DC),1(AC)
  Spectral selection = 1 .. 63
  Successive approximation = 0x10

  NOTE: Scan parsing doesn't support this SOF mode.

*** Marker: DHT (Define Huffman Table) (xFFC4) ***
  OFFSET: 0x00008A23
  Huffman table length = 42
  ----
  Destination ID = 1
  Class = 1 (AC Table)
    Codes of length 01 bits (000 total):
    Codes of length 02 bits (001 total): 01
    Codes of length 03 bits (002 total): 11 31
    Codes of length 04 bits (004 total): 21 50 81 A1
    Codes of length 05 bits (005 total): 00 40 41 71 B1
    Codes of length 06 bits (004 total): 30 91 C1 E1
    Codes of length 07 bits (003 total): 20 51 D1
    Codes of length 08 bits (001 total): 10
    Codes of length 09 bits (001 total): F1
    Codes of length 10 bits (001 total): F0
    Codes of length 11 bits (001 total): 61
```

```
    Codes of length 12 bits (000 total):
    Codes of length 13 bits (000 total):
    Codes of length 14 bits (000 total):
    Codes of length 15 bits (000 total):
    Codes of length 16 bits (000 total):
    Total number of codes: 023


*** Marker: SOS (Start of Scan) (xFFDA) ***
  OFFSET: 0x00008A4F
  Scan header length = 8
  Number of img components = 1
    Component[1]: selector=0x01, table=1(DC),1(AC)
  Spectral selection = 1 .. 63
  Successive approximation = 0x10

  NOTE: Scan parsing doesn't support this SOF mode.

*** Marker: DHT (Define Huffman Table) (xFFC4) ***
  OFFSET: 0x000094C2
  Huffman table length = 43
  ----
  Destination ID = 0
  Class = 1 (AC Table)
    Codes of length 01 bits (000 total):
    Codes of length 02 bits (002 total): 01 11
    Codes of length 03 bits (000 total):
    Codes of length 04 bits (004 total): 00 21 31 60
    Codes of length 05 bits (005 total): 41 50 51 71 81
    Codes of length 06 bits (003 total): 30 40 91
    Codes of length 07 bits (004 total): 61 A1 C1 F0
    Codes of length 08 bits (003 total): 20 D1 E1
    Codes of length 09 bits (001 total): 10
    Codes of length 10 bits (001 total): B1
```

```
    Codes of length 11 bits (001 total): F1
    Codes of length 12 bits (000 total):
    Codes of length 13 bits (000 total):
    Codes of length 14 bits (000 total):
    Codes of length 15 bits (000 total):
    Codes of length 16 bits (000 total):
    Total number of codes: 024


*** Marker: SOS (Start of Scan) (xFFDA) ***
  OFFSET: 0x000094EF
  Scan header length = 8
  Number of img components = 1
    Component[1]: selector=0x00, table=0(DC),0(AC)
  Spectral selection = 1 .. 63
  Successive approximation = 0x10

  NOTE: Scan parsing doesn't support this SOF mode.
```

Standard quality

Luminance quantization table: same as standard quality.

Chrominance quantization table: same as standard quality.

Chroma subsampling: same as standard quality.

Huffman tables:

```
*** Marker: DHT (Define Huffman Table) (xFFC4) ***
  OFFSET: 0x00000D49
  Huffman table length = 26
  ----
  Destination ID = 0
```

```
  Class = 0 (DC / Lossless Table)
    Codes of length 01 bits (001 total): 00
    Codes of length 02 bits (001 total): 05
    Codes of length 03 bits (001 total): 04
    Codes of length 04 bits (001 total): 01
    Codes of length 05 bits (001 total): 03
    Codes of length 06 bits (001 total): 02
    Codes of length 07 bits (001 total): 06
    Codes of length 08 bits (000 total):
    Codes of length 09 bits (000 total):
    Codes of length 10 bits (000 total):
    Codes of length 11 bits (000 total):
    Codes of length 12 bits (000 total):
    Codes of length 13 bits (000 total):
    Codes of length 14 bits (000 total):
    Codes of length 15 bits (000 total):
    Codes of length 16 bits (000 total):
    Total number of codes: 007


*** Marker: DHT (Define Huffman Table) (xFFC4) ***
  OFFSET: 0x00000D65
  Huffman table length = 26
  ----
  Destination ID = 1
  Class = 0 (DC / Lossless Table)
    Codes of length 01 bits (001 total): 00
    Codes of length 02 bits (000 total):
    Codes of length 03 bits (002 total): 04 06
    Codes of length 04 bits (003 total): 02 03 05
    Codes of length 05 bits (001 total): 01
    Codes of length 06 bits (000 total):
    Codes of length 07 bits (000 total):
    Codes of length 08 bits (000 total):
```

```
    Codes of length 09 bits (000 total):
    Codes of length 10 bits (000 total):
    Codes of length 11 bits (000 total):
    Codes of length 12 bits (000 total):
    Codes of length 13 bits (000 total):
    Codes of length 14 bits (000 total):
    Codes of length 15 bits (000 total):
    Codes of length 16 bits (000 total):
    Total number of codes: 007


*** Marker: DHT (Define Huffman Table) (xFFC4) ***
  OFFSET: 0x00000D81
  Huffman table length = 26
  ----
  Destination ID = 1
  Class = 0 (DC / Lossless Table)
    Codes of length 01 bits (001 total): 00
    Codes of length 02 bits (000 total):
    Codes of length 03 bits (002 total): 04 06
    Codes of length 04 bits (003 total): 02 03 05
    Codes of length 05 bits (001 total): 01
    Codes of length 06 bits (000 total):
    Codes of length 07 bits (000 total):
    Codes of length 08 bits (000 total):
    Codes of length 09 bits (000 total):
    Codes of length 10 bits (000 total):
    Codes of length 11 bits (000 total):
    Codes of length 12 bits (000 total):
    Codes of length 13 bits (000 total):
    Codes of length 14 bits (000 total):
    Codes of length 15 bits (000 total):
    Codes of length 16 bits (000 total):
    Total number of codes: 007
```

```
*** Marker: SOS (Start of Scan) (xFFDA) ***
  OFFSET: 0x00000D9D
  Scan header length = 12
  Number of img components = 3
    Component[1]: selector=0x00, table=0(DC),0(AC)
    Component[2]: selector=0x01, table=1(DC),1(AC)
    Component[3]: selector=0x02, table=1(DC),1(AC)
  Spectral selection = 0 .. 0
  Successive approximation = 0x01

  NOTE: Scan parsing doesn't support this SOF mode.

*** Marker: DHT (Define Huffman Table) (xFFC4) ***
  OFFSET: 0x00003962
  Huffman table length = 35
  ----
  Destination ID = 0
  Class = 1 (AC Table)
    Codes of length 01 bits (000 total):
    Codes of length 02 bits (001 total): 02
    Codes of length 03 bits (003 total): 01 03 60
    Codes of length 04 bits (004 total): 04 11 12 50
    Codes of length 05 bits (002 total): 21 40
    Codes of length 06 bits (003 total): 20 22 30
    Codes of length 07 bits (001 total): 00
    Codes of length 08 bits (001 total): 10
    Codes of length 09 bits (001 total): 31
    Codes of length 10 bits (000 total):
    Codes of length 11 bits (000 total):
    Codes of length 12 bits (000 total):
    Codes of length 13 bits (000 total):
    Codes of length 14 bits (000 total):
```

```
    Codes of length 15 bits (000 total):
    Codes of length 16 bits (000 total):
    Total number of codes: 016


*** Marker: SOS (Start of Scan) (xFFDA) ***
  OFFSET: 0x00003987
  Scan header length = 8
  Number of img components = 1
    Component[1]: selector=0x00, table=0(DC),0(AC)
  Spectral selection = 1 .. 5
  Successive approximation = 0x02

  NOTE: Scan parsing doesn't support this SOF mode.

*** Marker: DHT (Define Huffman Table) (xFFC4) ***
  OFFSET: 0x000043E3
  Huffman table length = 32
  ----
  Destination ID = 1
  Class = 1 (AC Table)
    Codes of length 01 bits (000 total):
    Codes of length 02 bits (002 total): 01 11
    Codes of length 03 bits (002 total): 03 50
    Codes of length 04 bits (002 total): 02 40
    Codes of length 05 bits (003 total): 21 30 41
    Codes of length 06 bits (001 total): 20
    Codes of length 07 bits (001 total): 00
    Codes of length 08 bits (001 total): 10
    Codes of length 09 bits (001 total): 70
    Codes of length 10 bits (000 total):
    Codes of length 11 bits (000 total):
    Codes of length 12 bits (000 total):
    Codes of length 13 bits (000 total):
```

```
    Codes of length 14 bits (000 total):
    Codes of length 15 bits (000 total):
    Codes of length 16 bits (000 total):
    Total number of codes: 013


*** Marker: SOS (Start of Scan) (xFFDA) ***
  OFFSET: 0x00004405
  Scan header length = 8
  Number of img components = 1
    Component[1]: selector=0x02, table=1(DC),1(AC)
  Spectral selection = 1 .. 63
  Successive approximation = 0x01

  NOTE: Scan parsing doesn't support this SOF mode.

*** Marker: DHT (Define Huffman Table) (xFFC4) ***
  OFFSET: 0x00004850
  Huffman table length = 42
  ----
  Destination ID = 1
  Class = 1 (AC Table)
    Codes of length 01 bits (000 total):
    Codes of length 02 bits (001 total): 03
    Codes of length 03 bits (002 total): 01 02
    Codes of length 04 bits (003 total): 05 13 50
    Codes of length 05 bits (008 total): 00 04 11 12 21 40 41 51
    Codes of length 06 bits (002 total): 30 61
    Codes of length 07 bits (003 total): 10 20 71
    Codes of length 08 bits (001 total): 32
    Codes of length 09 bits (001 total): 81
    Codes of length 10 bits (001 total): 31
    Codes of length 11 bits (001 total): 60
    Codes of length 12 bits (000 total):
```

```
    Codes of length 13 bits (000 total):
    Codes of length 14 bits (000 total):
    Codes of length 15 bits (000 total):
    Codes of length 16 bits (000 total):
    Total number of codes: 023


*** Marker: SOS (Start of Scan) (xFFDA) ***
  OFFSET: 0x0000487C
  Scan header length = 8
  Number of img components = 1
    Component[1]: selector=0x01, table=1(DC),1(AC)
  Spectral selection = 1 .. 63
  Successive approximation = 0x01

  NOTE: Scan parsing doesn't support this SOF mode.

*** Marker: DHT (Define Huffman Table) (xFFC4) ***
  OFFSET: 0x000052B7
  Huffman table length = 34
  ----
  Destination ID = 0
  Class = 1 (AC Table)
    Codes of length 01 bits (000 total):
    Codes of length 02 bits (002 total): 01 60
    Codes of length 03 bits (001 total): 11
    Codes of length 04 bits (004 total): 02 40 50 51
    Codes of length 05 bits (003 total): 00 30 71
    Codes of length 06 bits (001 total): 20
    Codes of length 07 bits (001 total): 61
    Codes of length 08 bits (001 total): 10
    Codes of length 09 bits (001 total): 12
    Codes of length 10 bits (001 total): 80
    Codes of length 11 bits (000 total):
```

```
    Codes of length 12 bits (000 total):
    Codes of length 13 bits (000 total):
    Codes of length 14 bits (000 total):
    Codes of length 15 bits (000 total):
    Codes of length 16 bits (000 total):
    Total number of codes: 015


*** Marker: SOS (Start of Scan) (xFFDA) ***
  OFFSET: 0x000052DB
  Scan header length = 8
  Number of img components = 1
    Component[1]: selector=0x00, table=0(DC),0(AC)
  Spectral selection = 6 .. 63
  Successive approximation = 0x02

  NOTE: Scan parsing doesn't support this SOF mode.

*** Marker: DHT (Define Huffman Table) (xFFC4) ***
  OFFSET: 0x0000586A
  Huffman table length = 38
  ----
  Destination ID = 0
  Class = 1 (AC Table)
    Codes of length 01 bits (001 total): 01
    Codes of length 02 bits (000 total):
    Codes of length 03 bits (000 total):
    Codes of length 04 bits (005 total): 11 21 41 60 61
    Codes of length 05 bits (004 total): 00 31 50 51
    Codes of length 06 bits (002 total): 30 40
    Codes of length 07 bits (002 total): 20 71
    Codes of length 08 bits (003 total): 81 A1 B1
    Codes of length 09 bits (001 total): 10
    Codes of length 10 bits (001 total): 70
```

```
    Codes of length 11 bits (000 total):
    Codes of length 12 bits (000 total):
    Codes of length 13 bits (000 total):
    Codes of length 14 bits (000 total):
    Codes of length 15 bits (000 total):
    Codes of length 16 bits (000 total):
    Total number of codes: 019


*** Marker: SOS (Start of Scan) (xFFDA) ***
  OFFSET: 0x00005892
  Scan header length = 8
  Number of img components = 1
    Component[1]: selector=0x00, table=0(DC),0(AC)
  Spectral selection = 1 .. 63
  Successive approximation = 0x21

  NOTE: Scan parsing doesn't support this SOF mode.

*** Marker: SOS (Start of Scan) (xFFDA) ***
  OFFSET: 0x000063AE
  Scan header length = 12
  Number of img components = 3
    Component[1]: selector=0x00, table=0(DC),0(AC)
    Component[2]: selector=0x01, table=1(DC),1(AC)
    Component[3]: selector=0x02, table=1(DC),1(AC)
  Spectral selection = 0 .. 0
  Successive approximation = 0x10

  NOTE: Scan parsing doesn't support this SOF mode.

*** Marker: DHT (Define Huffman Table) (xFFC4) ***
  OFFSET: 0x000085DC
  Huffman table length = 34
```

```
  ----
  Destination ID = 1
  Class = 1 (AC Table)
    Codes of length 01 bits (000 total):
    Codes of length 02 bits (002 total): 01 11
    Codes of length 03 bits (001 total): 50
    Codes of length 04 bits (004 total): 00 21 40 51
    Codes of length 05 bits (002 total): 30 61
    Codes of length 06 bits (003 total): 10 20 31
    Codes of length 07 bits (001 total): 81
    Codes of length 08 bits (001 total): 71
    Codes of length 09 bits (001 total): 70
    Codes of length 10 bits (000 total):
    Codes of length 11 bits (000 total):
    Codes of length 12 bits (000 total):
    Codes of length 13 bits (000 total):
    Codes of length 14 bits (000 total):
    Codes of length 15 bits (000 total):
    Codes of length 16 bits (000 total):
    Total number of codes: 015


*** Marker: SOS (Start of Scan) (xFFDA) ***
  OFFSET: 0x00008600
  Scan header length = 8
  Number of img components = 1
    Component[1]: selector=0x02, table=1(DC),1(AC)
  Spectral selection = 1 .. 63
  Successive approximation = 0x10

  NOTE: Scan parsing doesn't support this SOF mode.

*** Marker: DHT (Define Huffman Table) (xFFC4) ***
  OFFSET: 0x00008A23
```

```
  Huffman table length = 42
  ----
  Destination ID = 1
  Class = 1 (AC Table)
    Codes of length 01 bits (000 total):
    Codes of length 02 bits (001 total): 01
    Codes of length 03 bits (002 total): 11 31
    Codes of length 04 bits (004 total): 21 50 81 A1
    Codes of length 05 bits (005 total): 00 40 41 71 B1
    Codes of length 06 bits (004 total): 30 91 C1 E1
    Codes of length 07 bits (003 total): 20 51 D1
    Codes of length 08 bits (001 total): 10
    Codes of length 09 bits (001 total): F1
    Codes of length 10 bits (001 total): F0
    Codes of length 11 bits (001 total): 61
    Codes of length 12 bits (000 total):
    Codes of length 13 bits (000 total):
    Codes of length 14 bits (000 total):
    Codes of length 15 bits (000 total):
    Codes of length 16 bits (000 total):
    Total number of codes: 023


*** Marker: SOS (Start of Scan) (xFFDA) ***
  OFFSET: 0x00008A4F
  Scan header length = 8
  Number of img components = 1
    Component[1]: selector=0x01, table=1(DC),1(AC)
  Spectral selection = 1 .. 63
  Successive approximation = 0x10

  NOTE: Scan parsing doesn't support this SOF mode.

*** Marker: DHT (Define Huffman Table) (xFFC4) ***
```

```
OFFSET: 0x000094C2
Huffman table length = 43
----
Destination ID = 0
Class = 1 (AC Table)
  Codes of length 01 bits (000 total):
  Codes of length 02 bits (002 total): 01 11
  Codes of length 03 bits (000 total):
  Codes of length 04 bits (004 total): 00 21 31 60
  Codes of length 05 bits (005 total): 41 50 51 71 81
  Codes of length 06 bits (003 total): 30 40 91
  Codes of length 07 bits (004 total): 61 A1 C1 F0
  Codes of length 08 bits (003 total): 20 D1 E1
  Codes of length 09 bits (001 total): 10
  Codes of length 10 bits (001 total): B1
  Codes of length 11 bits (001 total): F1
  Codes of length 12 bits (000 total):
  Codes of length 13 bits (000 total):
  Codes of length 14 bits (000 total):
  Codes of length 15 bits (000 total):
  Codes of length 16 bits (000 total):
  Total number of codes: 024
```

12 Appendix 6 - Flickr compressions

2048 *

Luminance quantization table:

```
DQT, Row #0:   8   5   5   8  12  19  25  29
DQT, Row #1:   6   6   7   9  12  28  29  26
DQT, Row #2:   7   6   8  12  19  28  33  27
DQT, Row #3:   7   8  11  14  25  42  38  30
DQT, Row #4:   8  11  18  27  33  53  49  37
DQT, Row #5:  12  17  26  31  39  50  54  44
```

```
DQT, Row #6:  24  31  37  42  49  58  58  49
DQT, Row #7:  35  44  46  47  54  48  49  48
```

Chrominance quantization table:

```
DQT, Row #0:   8   8  12  23  48  48  48  48
DQT, Row #1:   8  10  12  32  48  48  48  48
DQT, Row #2:  12  12  27  48  48  48  48  48
DQT, Row #3:  23  32  48  48  48  48  48  48
DQT, Row #4:  48  48  48  48  48  48  48  48
DQT, Row #5:  48  48  48  48  48  48  48  48
DQT, Row #6:  48  48  48  48  48  48  48  48
DQT, Row #7:  48  48  48  48  48  48  48  48
```

Chroma subsampling:

```
Component[1]: ... (Subsamp 1 x 1), ... (Lum: Y)
Component[2]: ... (Subsamp 1 x 1), ... (Chrom: Cb)
Component[3]: ... (Subsamp 1 x 1), ... (Chrom: Cr)
```

Huffman tables:

```
  Destination ID = 0
  Class = 0 (DC / Lossless Table)
    Codes of length 01 bits (001 total): 00
    Codes of length 02 bits (001 total): 04
    Codes of length 03 bits (001 total): 07
    Codes of length 04 bits (001 total): 06
    Codes of length 05 bits (001 total): 05
    Codes of length 06 bits (001 total): 03
    Codes of length 07 bits (001 total): 02
    Codes of length 08 bits (001 total): 01
    Codes of length 09 bits (000 total):
    Codes of length 10 bits (000 total):
```

```
  Codes of length 11 bits (000 total):
  Codes of length 12 bits (000 total):
  Codes of length 13 bits (000 total):
  Codes of length 14 bits (000 total):
  Codes of length 15 bits (000 total):
  Codes of length 16 bits (000 total):
  Total number of codes: 008

Destination ID = 0
Class = 1 (AC Table)
  Codes of length 01 bits (001 total): 00
  Codes of length 02 bits (000 total):
  Codes of length 03 bits (002 total): 01 02
  Codes of length 04 bits (001 total): 03
  Codes of length 05 bits (001 total): 04
  Codes of length 06 bits (003 total): 05 06 11
  Codes of length 07 bits (009 total): 12 13 14 21 22 31 71 72 A1
  Codes of length 08 bits (007 total): 23 32 41 81 B1 C1 E1
  Codes of length 09 bits (004 total): 43 51 61 82
  Codes of length 10 bits (003 total): 42 91 F0
  Codes of length 11 bits (000 total):
  Codes of length 12 bits (003 total): 33 63 D1
  Codes of length 13 bits (000 total):
  Codes of length 14 bits (003 total): 52 53 A2
  Codes of length 15 bits (001 total): 73
  Codes of length 16 bits (000 total):
  Total number of codes: 038

Destination ID = 1
Class = 0 (DC / Lossless Table)
  Codes of length 01 bits (001 total): 00
  Codes of length 02 bits (001 total): 06
  Codes of length 03 bits (001 total): 07
  Codes of length 04 bits (001 total): 03
```

```
  Codes of length 05 bits (001 total): 05
  Codes of length 06 bits (001 total): 04
  Codes of length 07 bits (001 total): 08
  Codes of length 08 bits (001 total): 02
  Codes of length 09 bits (001 total): 01
  Codes of length 10 bits (000 total):
  Codes of length 11 bits (000 total):
  Codes of length 12 bits (000 total):
  Codes of length 13 bits (000 total):
  Codes of length 14 bits (000 total):
  Codes of length 15 bits (000 total):
  Codes of length 16 bits (000 total):
  Total number of codes: 009

Destination ID = 1
Class = 1 (AC Table)
  Codes of length 01 bits (001 total): 00
  Codes of length 02 bits (000 total):
  Codes of length 03 bits (001 total): 01
  Codes of length 04 bits (002 total): 02 03
  Codes of length 05 bits (004 total): 04 05 11 21
  Codes of length 06 bits (003 total): 12 71 81
  Codes of length 07 bits (007 total): 06 14 22 31 41 91 A1
  Codes of length 08 bits (003 total): 07 13 51
  Codes of length 09 bits (003 total): 32 72 C1
  Codes of length 10 bits (004 total): 15 42 61 82
  Codes of length 11 bits (003 total): 62 B1 D1
  Codes of length 12 bits (001 total): A2
  Codes of length 13 bits (001 total): 33
  Codes of length 14 bits (001 total): E1
  Codes of length 15 bits (000 total):
  Codes of length 16 bits (000 total):
  Total number of codes: 034
```

1600 *

Luminance quantization table:

```
DQT, Row #0:   4   3   1   4   6   9  12  14
DQT, Row #1:   3   3   3   5   6  14  14  13
DQT, Row #2:   3   3   4   6   9  13  16  13
DQT, Row #3:   3   4   5   7  12  20  18  14
DQT, Row #4:   4   5   9  13  16  25  24  18
DQT, Row #5:   6   8  13  15  19  24  26  21
DQT, Row #6:  11  15  18  20  24  28  28  23
DQT, Row #7:  17  21  22  23  26  23  24  23
```

Chrominance quantization table:

```
DQT, Row #0:   4   4   6  11  23  23  23  23
DQT, Row #1:   4   5   6  15  23  23  23  23
DQT, Row #2:   6   6  13  23  23  23  23  23
DQT, Row #3:  11  15  23  23  23  23  23  23
DQT, Row #4:  23  23  23  23  23  23  23  23
DQT, Row #5:  23  23  23  23  23  23  23  23
DQT, Row #6:  23  23  23  23  23  23  23  23
DQT, Row #7:  23  23  23  23  23  23  23  23
```

Chroma subsampling:

```
Component[1]: ... (Subsamp 1 x 1), ... (Lum: Y)
Component[2]: ... (Subsamp 1 x 1), ... (Chrom: Cb)
Component[3]: ... (Subsamp 1 x 1), ... (Chrom: Cr)
```

Huffman tables:

```
OFFSET: 0x000000CF
Huffman table length = 28
----
```

```
  Destination ID = 0
  Class = 0 (DC / Lossless Table)
    Codes of length 01 bits (001 total): 00
    Codes of length 02 bits (000 total):
    Codes of length 03 bits (002 total): 01 07
    Codes of length 04 bits (002 total): 05 06
    Codes of length 05 bits (003 total): 02 04 08
    Codes of length 06 bits (001 total): 03
    Codes of length 07 bits (000 total):
    Codes of length 08 bits (000 total):
    Codes of length 09 bits (000 total):
    Codes of length 10 bits (000 total):
    Codes of length 11 bits (000 total):
    Codes of length 12 bits (000 total):
    Codes of length 13 bits (000 total):
    Codes of length 14 bits (000 total):
    Codes of length 15 bits (000 total):
    Codes of length 16 bits (000 total):
    Total number of codes: 009


*** Marker: DHT (Define Huffman Table) (xFFC4) ***
  OFFSET: 0x000000ED
  Huffman table length = 69
  ----
  Destination ID = 0
  Class = 1 (AC Table)
    Codes of length 01 bits (001 total): 00
    Codes of length 02 bits (000 total):
    Codes of length 03 bits (001 total): 02
    Codes of length 04 bits (002 total): 01 03
    Codes of length 05 bits (002 total): 04 05
    Codes of length 06 bits (004 total): 06 07 11 12
    Codes of length 07 bits (008 total): 13 14 15 21 22 31 72 81
```

```
    Codes of length 08 bits (010 total): 23 32 41 51 61 71 73 92 A2 B2
    Codes of length 09 bits (009 total): 33 42 62 82 91 A1 B1 C2 D1
    Codes of length 10 bits (005 total): 24 44 83 C1 E2
    Codes of length 11 bits (000 total):
    Codes of length 12 bits (001 total): 34
    Codes of length 13 bits (004 total): 43 63 64 F0
    Codes of length 14 bits (003 total): 52 53 D2
    Codes of length 15 bits (000 total):
    Codes of length 16 bits (000 total):
    Total number of codes: 050


*** Marker: DHT (Define Huffman Table) (xFFC4) ***
  OFFSET: 0x00000134
  Huffman table length = 29
  ----
  Destination ID = 1
  Class = 0 (DC / Lossless Table)
    Codes of length 01 bits (001 total): 00
    Codes of length 02 bits (000 total):
    Codes of length 03 bits (001 total): 06
    Codes of length 04 bits (004 total): 01 05 07 08
    Codes of length 05 bits (003 total): 03 04 09
    Codes of length 06 bits (001 total): 02
    Codes of length 07 bits (000 total):
    Codes of length 08 bits (000 total):
    Codes of length 09 bits (000 total):
    Codes of length 10 bits (000 total):
    Codes of length 11 bits (000 total):
    Codes of length 12 bits (000 total):
    Codes of length 13 bits (000 total):
    Codes of length 14 bits (000 total):
    Codes of length 15 bits (000 total):
    Codes of length 16 bits (000 total):
```

```
    Total number of codes: 010


*** Marker: DHT (Define Huffman Table) (xFFC4) ***
  OFFSET: 0x00000153
  Huffman table length = 64
  ----
  Destination ID = 1
  Class = 1 (AC Table)
    Codes of length 01 bits (001 total): 00
    Codes of length 02 bits (000 total):
    Codes of length 03 bits (001 total): 01
    Codes of length 04 bits (002 total): 02 03
    Codes of length 05 bits (003 total): 04 05 11
    Codes of length 06 bits (004 total): 06 21 31 81
    Codes of length 07 bits (007 total): 12 13 22 41 71 91 A1
    Codes of length 08 bits (005 total): 07 14 51 72 82
    Codes of length 09 bits (007 total): 08 15 32 61 92 B1 E1
    Codes of length 10 bits (004 total): 16 42 C1 D1
    Codes of length 11 bits (002 total): A2 F0
    Codes of length 12 bits (003 total): 23 43 F1
    Codes of length 13 bits (001 total): 62
    Codes of length 14 bits (000 total):
    Codes of length 15 bits (002 total): 73 B2
    Codes of length 16 bits (003 total): 34 C2 D2
    Total number of codes: 045
```

1024 *

Luminance quantization table:

```
    DQT, Row #0:   4   3   3   4   7  11  14  17
    DQT, Row #1:   3   3   4   6   7  16  17  15
    DQT, Row #2:   4   4   4   7  11  16  20  15
    DQT, Row #3:   4   5   6   8  14  24  22  17
```

```
DQT, Row #4:   5   6  10  15  19  31  29  21
DQT, Row #5:   7  10  15  18  22  29  32  26
DQT, Row #6:  14  18  22  24  29  34  34  28
DQT, Row #7:  20  26  27  28  31  28  29  28
```

Chrominance quantization table:

```
DQT, Row #0:   5   5   7  13  28  28  28  28
DQT, Row #1:   5   6   7  18  28  28  28  28
DQT, Row #2:   7   7  15  28  28  28  28  28
DQT, Row #3:  13  18  28  28  28  28  28  28
DQT, Row #4:  28  28  28  28  28  28  28  28
DQT, Row #5:  28  28  28  28  28  28  28  28
DQT, Row #6:  28  28  28  28  28  28  28  28
DQT, Row #7:  28  28  28  28  28  28  28  28
```

Chroma subsampling:

```
Component[1]: ... (Subsamp 1 x 1), ... (Lum: Y)
Component[2]: ... (Subsamp 1 x 1), ... (Chrom: Cb)
Component[3]: ... (Subsamp 1 x 1), ... (Chrom: Cr)
```

Huffman tables:

```
OFFSET: 0x000000CF
Huffman table length = 28
----
Destination ID = 0
Class = 0 (DC / Lossless Table)
  Codes of length 01 bits (001 total): 00
  Codes of length 02 bits (001 total): 07
  Codes of length 03 bits (000 total):
  Codes of length 04 bits (002 total): 05 06
  Codes of length 05 bits (003 total): 03 04 08
```

```
    Codes of length 06 bits (001 total): 01
    Codes of length 07 bits (001 total): 02
    Codes of length 08 bits (000 total):
    Codes of length 09 bits (000 total):
    Codes of length 10 bits (000 total):
    Codes of length 11 bits (000 total):
    Codes of length 12 bits (000 total):
    Codes of length 13 bits (000 total):
    Codes of length 14 bits (000 total):
    Codes of length 15 bits (000 total):
    Codes of length 16 bits (000 total):
    Total number of codes: 009


*** Marker: DHT (Define Huffman Table) (xFFC4) ***
  OFFSET: 0x000000ED
  Huffman table length = 64
  ----
  Destination ID = 0
  Class = 1 (AC Table)
    Codes of length 01 bits (001 total): 00
    Codes of length 02 bits (000 total):
    Codes of length 03 bits (001 total): 02
    Codes of length 04 bits (003 total): 01 03 04
    Codes of length 05 bits (001 total): 05
    Codes of length 06 bits (003 total): 06 11 21
    Codes of length 07 bits (007 total): 07 12 13 15 31 72 A1
    Codes of length 08 bits (010 total): 14 22 32 51 61 71 81 91 A2 B1
    Codes of length 09 bits (003 total): 23 41 82
    Codes of length 10 bits (008 total): 24 33 42 62 C1 C2 D1 E2
    Codes of length 11 bits (002 total): 43 F0
    Codes of length 12 bits (003 total): 64 B2 E1
    Codes of length 13 bits (001 total): 92
    Codes of length 14 bits (001 total): 34
```

```
    Codes of length 15 bits (001 total): 52
    Codes of length 16 bits (000 total):
    Total number of codes: 045


*** Marker: DHT (Define Huffman Table) (xFFC4) ***
  OFFSET: 0x0000012F
  Huffman table length = 28
  ----
  Destination ID = 1
  Class = 0 (DC / Lossless Table)
    Codes of length 01 bits (001 total): 00
    Codes of length 02 bits (000 total):
    Codes of length 03 bits (002 total): 06 08
    Codes of length 04 bits (003 total): 01 05 07
    Codes of length 05 bits (001 total): 04
    Codes of length 06 bits (001 total): 03
    Codes of length 07 bits (001 total): 02
    Codes of length 08 bits (000 total):
    Codes of length 09 bits (000 total):
    Codes of length 10 bits (000 total):
    Codes of length 11 bits (000 total):
    Codes of length 12 bits (000 total):
    Codes of length 13 bits (000 total):
    Codes of length 14 bits (000 total):
    Codes of length 15 bits (000 total):
    Codes of length 16 bits (000 total):
    Total number of codes: 009


*** Marker: DHT (Define Huffman Table) (xFFC4) ***
  OFFSET: 0x0000014D
  Huffman table length = 58
  ----
```

```
Destination ID = 1
Class = 1 (AC Table)
  Codes of length 01 bits (001 total): 00
  Codes of length 02 bits (000 total):
  Codes of length 03 bits (001 total): 01
  Codes of length 04 bits (002 total): 02 03
  Codes of length 05 bits (003 total): 04 05 11
  Codes of length 06 bits (003 total): 21 31 81
  Codes of length 07 bits (009 total): 06 07 12 13 22 41 71 91 A1
  Codes of length 08 bits (006 total): 14 15 51 72 82 E1
  Codes of length 09 bits (006 total): 32 61 92 B1 C1 D1
  Codes of length 10 bits (002 total): 23 F0
  Codes of length 11 bits (002 total): 43 F1
  Codes of length 12 bits (003 total): 42 A2 B2
  Codes of length 13 bits (001 total): 34
  Codes of length 14 bits (000 total):
  Codes of length 15 bits (000 total):
  Codes of length 16 bits (000 total):
  Total number of codes: 039
```

800 * Luminance quantization table:

```
DQT, Row #0:   4   3   3   4   6  10  13  15
DQT, Row #1:   3   3   3   5   6  15  15  13
DQT, Row #2:   3   3   4   6  10  14  17  14
DQT, Row #3:   3   4   6   7  13  22  20  15
DQT, Row #4:   4   6   9  14  17  27  25  19
DQT, Row #5:   6   9  13  16  20  26  28  23
DQT, Row #6:  12  16  19  22  25  30  30  25
DQT, Row #7:  18  23  24  24  28  25  25  25
```

Chrominance quantization table:

```
DQT, Row #0:   4   4   6  12  25  25  25  25
```

```
DQT, Row #1:   4   5   6  16  25  25  25  25
DQT, Row #2:   6   6  14  25  25  25  25  25
DQT, Row #3:  12  16  25  25  25  25  25  25
DQT, Row #4:  25  25  25  25  25  25  25  25
DQT, Row #5:  25  25  25  25  25  25  25  25
DQT, Row #6:  25  25  25  25  25  25  25  25
DQT, Row #7:  25  25  25  25  25  25  25  25
```

Chroma subsampling:

```
Component[1]: ... (Subsamp 1 x 1), ... (Lum: Y)
Component[2]: ... (Subsamp 1 x 1), ... (Chrom: Cb)
Component[3]: ... (Subsamp 1 x 1), ... (Chrom: Cr)
```

Huffman tables:

```
OFFSET: 0x000000CF
Huffman table length = 28
----
Destination ID = 0
Class = 0 (DC / Lossless Table)
  Codes of length 01 bits (001 total): 00
  Codes of length 02 bits (000 total):
  Codes of length 03 bits (002 total): 01 07
  Codes of length 04 bits (002 total): 05 06
  Codes of length 05 bits (003 total): 02 04 08
  Codes of length 06 bits (001 total): 03
  Codes of length 07 bits (000 total):
  Codes of length 08 bits (000 total):
  Codes of length 09 bits (000 total):
  Codes of length 10 bits (000 total):
  Codes of length 11 bits (000 total):
  Codes of length 12 bits (000 total):
  Codes of length 13 bits (000 total):
```

```
    Codes of length 14 bits (000 total):
    Codes of length 15 bits (000 total):
    Codes of length 16 bits (000 total):
    Total number of codes: 009


*** Marker: DHT (Define Huffman Table) (xFFC4) ***
  OFFSET: 0x000000ED
  Huffman table length = 64
  ----
  Destination ID = 0
  Class = 1 (AC Table)
    Codes of length 01 bits (001 total): 00
    Codes of length 02 bits (000 total):
    Codes of length 03 bits (001 total): 01
    Codes of length 04 bits (003 total): 02 03 04
    Codes of length 05 bits (002 total): 05 11
    Codes of length 06 bits (002 total): 12 81
    Codes of length 07 bits (005 total): 06 07 21 22 91
    Codes of length 08 bits (008 total): 13 14 31 41 51 82 92 A1
    Codes of length 09 bits (008 total): 15 32 52 61 62 63 71 73
    Codes of length 10 bits (006 total): 23 33 42 72 A2 B1
    Codes of length 11 bits (001 total): 53
    Codes of length 12 bits (004 total): 83 B2 C1 C2
    Codes of length 13 bits (003 total): 24 43 D1
    Codes of length 14 bits (001 total): 44
    Codes of length 15 bits (000 total):
    Codes of length 16 bits (000 total):
    Total number of codes: 045


*** Marker: DHT (Define Huffman Table) (xFFC4) ***
  OFFSET: 0x0000012F
  Huffman table length = 29
```

```
  ----
  Destination ID = 1
  Class = 0 (DC / Lossless Table)
    Codes of length 01 bits (001 total): 00
    Codes of length 02 bits (000 total):
    Codes of length 03 bits (001 total): 06
    Codes of length 04 bits (004 total): 01 05 07 08
    Codes of length 05 bits (003 total): 03 04 09
    Codes of length 06 bits (001 total): 02
    Codes of length 07 bits (000 total):
    Codes of length 08 bits (000 total):
    Codes of length 09 bits (000 total):
    Codes of length 10 bits (000 total):
    Codes of length 11 bits (000 total):
    Codes of length 12 bits (000 total):
    Codes of length 13 bits (000 total):
    Codes of length 14 bits (000 total):
    Codes of length 15 bits (000 total):
    Codes of length 16 bits (000 total):
    Total number of codes: 010


*** Marker: DHT (Define Huffman Table) (xFFC4) ***
  OFFSET: 0x0000014E
  Huffman table length = 64
  ----
  Destination ID = 1
  Class = 1 (AC Table)
    Codes of length 01 bits (001 total): 00
    Codes of length 02 bits (000 total):
    Codes of length 03 bits (001 total): 01
    Codes of length 04 bits (002 total): 02 03
    Codes of length 05 bits (003 total): 04 05 11
    Codes of length 06 bits (004 total): 06 21 31 81
```

```
Codes of length 07 bits (006 total): 12 13 22 71 91 A1
Codes of length 08 bits (007 total): 07 14 41 51 61 72 82
Codes of length 09 bits (007 total): 08 15 32 B1 C1 D1 E1
Codes of length 10 bits (004 total): 16 42 73 92
Codes of length 11 bits (001 total): F0
Codes of length 12 bits (004 total): 43 52 62 A2
Codes of length 13 bits (003 total): 23 83 B2
Codes of length 14 bits (001 total): 34
Codes of length 15 bits (001 total): 33
Codes of length 16 bits (000 total):
Total number of codes: 045
```

500 * Luminance quantization table:

```
DQT, Row #0:   3   1   1   3   2   7   9  11
DQT, Row #1:   1   1   1   2   2   5  11   9
DQT, Row #2:   1   1   3   2   7   5   6   9
DQT, Row #3:   1   3   2   5   9  15  14  11
DQT, Row #4:   3   2   3   9  12  19  18  13
DQT, Row #5:   2   3   9  11  14  18  20  16
DQT, Row #6:   9  11  13  15  18  21  21  18
DQT, Row #7:  13  16  17  17  20  17  18  17
```

Chrominance quantization table:

```
DQT, Row #0:   3   3   2   4  17  17  17  17
DQT, Row #1:   3   2   2  11  17  17  17  17
DQT, Row #2:   2   2   9  17  17  17  17  17
DQT, Row #3:   4  11  17  17  17  17  17  17
DQT, Row #4:  17  17  17  17  17  17  17  17
DQT, Row #5:  17  17  17  17  17  17  17  17
DQT, Row #6:  17  17  17  17  17  17  17  17
DQT, Row #7:  17  17  17  17  17  17  17  17
```

Chroma subsampling:

```
    Component[1]: ... (Subsamp 1 x 1), ... (Lum: Y)
    Component[2]: ... (Subsamp 1 x 1), ... (Chrom: Cb)
    Component[3]: ... (Subsamp 1 x 1), ... (Chrom: Cr)
```

Huffman tables:

```
  OFFSET: 0x000000CF
  Huffman table length = 28
  ----
  Destination ID = 0
  Class = 0 (DC / Lossless Table)
    Codes of length 01 bits (001 total): 00
    Codes of length 02 bits (000 total):
    Codes of length 03 bits (002 total): 07 08
    Codes of length 04 bits (002 total): 01 06
    Codes of length 05 bits (003 total): 02 04 05
    Codes of length 06 bits (001 total): 03
    Codes of length 07 bits (000 total):
    Codes of length 08 bits (000 total):
    Codes of length 09 bits (000 total):
    Codes of length 10 bits (000 total):
    Codes of length 11 bits (000 total):
    Codes of length 12 bits (000 total):
    Codes of length 13 bits (000 total):
    Codes of length 14 bits (000 total):
    Codes of length 15 bits (000 total):
    Codes of length 16 bits (000 total):
    Total number of codes: 009


*** Marker: DHT (Define Huffman Table) (xFFC4) ***
  OFFSET: 0x000000ED
  Huffman table length = 73
  ----
```

```
  Destination ID = 0
  Class = 1 (AC Table)
    Codes of length 01 bits (000 total):
    Codes of length 02 bits (001 total): 00
    Codes of length 03 bits (003 total): 01 02 03
    Codes of length 04 bits (002 total): 04 05
    Codes of length 05 bits (003 total): 06 07 11
    Codes of length 06 bits (004 total): 08 12 22 81
    Codes of length 07 bits (004 total): 13 31 82 91
    Codes of length 08 bits (010 total): 14 15 16 21 32 42 51 52 63 92
    Codes of length 09 bits (007 total): 41 61 62 71 73 93 A2
    Codes of length 10 bits (007 total): 17 23 43 72 83 A1 B1
    Codes of length 11 bits (002 total): 33 A3
    Codes of length 12 bits (004 total): 34 35 53 B3
    Codes of length 13 bits (007 total): 24 25 65 94 B2 C1 C2
    Codes of length 14 bits (000 total):
    Codes of length 15 bits (000 total):
    Codes of length 16 bits (000 total):
    Total number of codes: 054


*** Marker: DHT (Define Huffman Table) (xFFC4) ***
  OFFSET: 0x00000138
  Huffman table length = 29
  ----
  Destination ID = 1
  Class = 0 (DC / Lossless Table)
    Codes of length 01 bits (001 total): 00
    Codes of length 02 bits (000 total):
    Codes of length 03 bits (001 total): 07
    Codes of length 04 bits (004 total): 05 06 08 09
    Codes of length 05 bits (003 total): 01 02 04
    Codes of length 06 bits (001 total): 03
    Codes of length 07 bits (000 total):
```

```
    Codes of length 08 bits (000 total):
    Codes of length 09 bits (000 total):
    Codes of length 10 bits (000 total):
    Codes of length 11 bits (000 total):
    Codes of length 12 bits (000 total):
    Codes of length 13 bits (000 total):
    Codes of length 14 bits (000 total):
    Codes of length 15 bits (000 total):
    Codes of length 16 bits (000 total):
    Total number of codes: 010


*** Marker: DHT (Define Huffman Table) (xFFC4) ***
  OFFSET: 0x00000157
  Huffman table length = 71
  ----
  Destination ID = 1
  Class = 1 (AC Table)
    Codes of length 01 bits (001 total): 00
    Codes of length 02 bits (000 total):
    Codes of length 03 bits (001 total): 01
    Codes of length 04 bits (002 total): 02 03
    Codes of length 05 bits (004 total): 04 05 06 11
    Codes of length 06 bits (002 total): 07 21
    Codes of length 07 bits (005 total): 12 22 31 81 82
    Codes of length 08 bits (007 total): 08 13 14 71 73 91 A1
    Codes of length 09 bits (009 total): 15 41 51 61 72 83 92 B1 E1
    Codes of length 10 bits (006 total): 16 23 32 42 C1 D1
    Codes of length 11 bits (005 total): 33 62 93 A2 F0
    Codes of length 12 bits (004 total): 34 36 44 B3
    Codes of length 13 bits (001 total): 24
    Codes of length 14 bits (005 total): 17 52 84 B2 F1
    Codes of length 15 bits (000 total):
    Codes of length 16 bits (000 total):
```

```
        Total number of codes: 052
```

13 Appendix 7 - GIMP progressive JPEG: Huffman tables

```
*** Marker: DHT (Define Huffman Table) (xFFC4) ***
  OFFSET: 0x00000D0B
  Huffman table length = 26
  ----
  Destination ID = 0
  Class = 0 (DC / Lossless Table)
    Codes of length 01 bits (001 total): 00
    Codes of length 02 bits (001 total): 01
    Codes of length 03 bits (001 total): 02
    Codes of length 04 bits (001 total): 03
    Codes of length 05 bits (001 total): 04
    Codes of length 06 bits (001 total): 05
    Codes of length 07 bits (001 total): 06
    Codes of length 08 bits (000 total):
    Codes of length 09 bits (000 total):
    Codes of length 10 bits (000 total):
    Codes of length 11 bits (000 total):
    Codes of length 12 bits (000 total):
    Codes of length 13 bits (000 total):
    Codes of length 14 bits (000 total):
    Codes of length 15 bits (000 total):
    Codes of length 16 bits (000 total):
    Total number of codes: 007


*** Marker: DHT (Define Huffman Table) (xFFC4) ***
  OFFSET: 0x00000D27
  Huffman table length = 24
```

```
  ----
  Destination ID = 1
  Class = 0 (DC / Lossless Table)
    Codes of length 01 bits (001 total): 00
    Codes of length 02 bits (001 total): 01
    Codes of length 03 bits (001 total): 02
    Codes of length 04 bits (001 total): 03
    Codes of length 05 bits (001 total): 04
    Codes of length 06 bits (000 total):
    Codes of length 07 bits (000 total):
    Codes of length 08 bits (000 total):
    Codes of length 09 bits (000 total):
    Codes of length 10 bits (000 total):
    Codes of length 11 bits (000 total):
    Codes of length 12 bits (000 total):
    Codes of length 13 bits (000 total):
    Codes of length 14 bits (000 total):
    Codes of length 15 bits (000 total):
    Codes of length 16 bits (000 total):
    Total number of codes: 005


*** Marker: SOS (Start of Scan) (xFFDA) ***
  OFFSET: 0x00000D41
  Scan header length = 12
  Number of img components = 3
    Component[1]: selector=0x01, table=0(DC),0(AC)
    Component[2]: selector=0x02, table=1(DC),0(AC)
    Component[3]: selector=0x03, table=1(DC),0(AC)
  Spectral selection = 0 .. 0
  Successive approximation = 0x01

  NOTE: Scan parsing doesn't support this SOF mode.
```

```
*** Marker: DHT (Define Huffman Table) (xFFC4) ***
  OFFSET: 0x0002D93E
  Huffman table length = 52
  ----
  Destination ID = 0
  Class = 1 (AC Table)
    Codes of length 01 bits (000 total):
    Codes of length 02 bits (002 total): 00 01
    Codes of length 03 bits (002 total): 02 11
    Codes of length 04 bits (002 total): 10 12
    Codes of length 05 bits (002 total): 03 21
    Codes of length 06 bits (002 total): 20 31
    Codes of length 07 bits (001 total): 41
    Codes of length 08 bits (004 total): 13 22 30 32
    Codes of length 09 bits (002 total): 04 40
    Codes of length 10 bits (001 total): 50
    Codes of length 11 bits (004 total): 14 42 60 90
    Codes of length 12 bits (003 total): 33 70 80
    Codes of length 13 bits (001 total): 23
    Codes of length 14 bits (000 total):
    Codes of length 15 bits (000 total):
    Codes of length 16 bits (007 total): 43 A0 B0 C0 24 D0 E0
    Total number of codes: 033


*** Marker: SOS (Start of Scan) (xFFDA) ***
  OFFSET: 0x0002D974
  Scan header length = 8
  Number of img components = 1
    Component[1]: selector=0x01, table=0(DC),0(AC)
  Spectral selection = 1 .. 5
  Successive approximation = 0x02

  NOTE: Scan parsing doesn't support this SOF mode.
```

```
*** Marker: DHT (Define Huffman Table) (xFFC4) ***
  OFFSET: 0x00041C3C
  Huffman table length = 42
  ----
  Destination ID = 1
  Class = 1 (AC Table)
    Codes of length 01 bits (000 total):
    Codes of length 02 bits (002 total): 00 11
    Codes of length 03 bits (001 total): 01
    Codes of length 04 bits (003 total): 10 20 30
    Codes of length 05 bits (004 total): 12 40 50 60
    Codes of length 06 bits (003 total): 02 70 80
    Codes of length 07 bits (001 total): 90
    Codes of length 08 bits (001 total): 21
    Codes of length 09 bits (001 total): 31
    Codes of length 10 bits (000 total):
    Codes of length 11 bits (002 total): A0 D0
    Codes of length 12 bits (002 total): 13 E0
    Codes of length 13 bits (003 total): 03 41 51
    Codes of length 14 bits (000 total):
    Codes of length 15 bits (000 total):
    Codes of length 16 bits (000 total):
    Total number of codes: 023


*** Marker: SOS (Start of Scan) (xFFDA) ***
  OFFSET: 0x00041C68
  Scan header length = 8
  Number of img components = 1
    Component[1]: selector=0x03, table=0(DC),1(AC)
  Spectral selection = 1 .. 63
  Successive approximation = 0x01
```

```
  NOTE: Scan parsing doesn't support this SOF mode.

*** Marker: DHT (Define Huffman Table) (xFFC4) ***
  OFFSET: 0x00042BF5
  Huffman table length = 43
  ----
  Destination ID = 1
  Class = 1 (AC Table)
    Codes of length 01 bits (000 total):
    Codes of length 02 bits (002 total): 00 11
    Codes of length 03 bits (002 total): 01 10
    Codes of length 04 bits (002 total): 20 30
    Codes of length 05 bits (002 total): 12 40
    Codes of length 06 bits (001 total): 50
    Codes of length 07 bits (005 total): 02 21 60 70 80
    Codes of length 08 bits (001 total): 90
    Codes of length 09 bits (001 total): 31
    Codes of length 10 bits (000 total):
    Codes of length 11 bits (002 total): 41 A0
    Codes of length 12 bits (003 total): 03 D0 E0
    Codes of length 13 bits (000 total):
    Codes of length 14 bits (003 total): 13 51 B0
    Codes of length 15 bits (000 total):
    Codes of length 16 bits (000 total):
    Total number of codes: 024


*** Marker: SOS (Start of Scan) (xFFDA) ***
  OFFSET: 0x00042C22
  Scan header length = 8
  Number of img components = 1
    Component[1]: selector=0x02, table=0(DC),1(AC)
  Spectral selection = 1 .. 63
  Successive approximation = 0x01
```

```
  NOTE: Scan parsing doesn't support this SOF mode.

*** Marker: DHT (Define Huffman Table) (xFFC4) ***
  OFFSET: 0x000449AD
  Huffman table length = 45
  ----
  Destination ID = 0
  Class = 1 (AC Table)
    Codes of length 01 bits (000 total):
    Codes of length 02 bits (001 total): 21
    Codes of length 03 bits (003 total): 00 11 31
    Codes of length 04 bits (004 total): 01 10 20 30
    Codes of length 05 bits (002 total): 40 50
    Codes of length 06 bits (001 total): 60
    Codes of length 07 bits (003 total): 02 22 70
    Codes of length 08 bits (004 total): 12 32 61 80
    Codes of length 09 bits (003 total): 41 51 90
    Codes of length 10 bits (001 total): 33
    Codes of length 11 bits (001 total): A0
    Codes of length 12 bits (001 total): E0
    Codes of length 13 bits (001 total): B0
    Codes of length 14 bits (001 total): D0
    Codes of length 15 bits (000 total):
    Codes of length 16 bits (000 total):
    Total number of codes: 026


*** Marker: SOS (Start of Scan) (xFFDA) ***
  OFFSET: 0x000449DC
  Scan header length = 8
  Number of img components = 1
    Component[1]: selector=0x01, table=0(DC),0(AC)
  Spectral selection = 6 .. 63
```

```
  Successive approximation = 0x02

  NOTE: Scan parsing doesn't support this SOF mode.

*** Marker: DHT (Define Huffman Table) (xFFC4) ***
  OFFSET: 0x00046827
  Huffman table length = 45
  ----
  Destination ID = 0
  Class = 1 (AC Table)
    Codes of length 01 bits (000 total):
    Codes of length 02 bits (003 total): 00 01 11
    Codes of length 03 bits (000 total):
    Codes of length 04 bits (002 total): 21 31
    Codes of length 05 bits (002 total): 10 41
    Codes of length 06 bits (002 total): 20 51
    Codes of length 07 bits (002 total): 30 61
    Codes of length 08 bits (003 total): 40 50 71
    Codes of length 09 bits (001 total): 60
    Codes of length 10 bits (001 total): 81
    Codes of length 11 bits (000 total):
    Codes of length 12 bits (002 total): 90 91
    Codes of length 13 bits (003 total): 70 A1 B1
    Codes of length 14 bits (000 total):
    Codes of length 15 bits (002 total): 80 D1
    Codes of length 16 bits (003 total): C1 F1 F0
    Total number of codes: 026


*** Marker: SOS (Start of Scan) (xFFDA) ***
  OFFSET: 0x00046856
  Scan header length = 8
  Number of img components = 1
    Component[1]: selector=0x01, table=0(DC),0(AC)
```

```
  Spectral selection = 1 .. 63
  Successive approximation = 0x21

  NOTE: Scan parsing doesn't support this SOF mode.

*** Marker: SOS (Start of Scan) (xFFDA) ***
  OFFSET: 0x00069752
  Scan header length = 12
  Number of img components = 3
    Component[1]: selector=0x01, table=0(DC),0(AC)
    Component[2]: selector=0x02, table=0(DC),0(AC)
    Component[3]: selector=0x03, table=0(DC),0(AC)
  Spectral selection = 0 .. 0
  Successive approximation = 0x10

  NOTE: Scan parsing doesn't support this SOF mode.

*** Marker: DHT (Define Huffman Table) (xFFC4) ***
  OFFSET: 0x00082EEF
  Huffman table length = 41
  ----
  Destination ID = 1
  Class = 1 (AC Table)
    Codes of length 01 bits (000 total):
    Codes of length 02 bits (003 total): 00 01 11
    Codes of length 03 bits (001 total): 10
    Codes of length 04 bits (001 total): 20
    Codes of length 05 bits (001 total): 30
    Codes of length 06 bits (000 total):
    Codes of length 07 bits (002 total): 21 40
    Codes of length 08 bits (003 total): 31 50 60
    Codes of length 09 bits (000 total):
    Codes of length 10 bits (003 total): 41 70 90
    Codes of length 11 bits (001 total): 80
```

```
    Codes of length 12 bits (001 total): 51
    Codes of length 13 bits (001 total): A0
    Codes of length 14 bits (000 total):
    Codes of length 15 bits (002 total): 61 B0
    Codes of length 16 bits (003 total): C0 D0 E0
    Total number of codes: 022


*** Marker: SOS (Start of Scan) (xFFDA) ***
  OFFSET: 0x00082F1A
  Scan header length = 8
  Number of img components = 1
    Component[1]: selector=0x03, table=0(DC),1(AC)
  Spectral selection = 1 .. 63
  Successive approximation = 0x10

  NOTE: Scan parsing doesn't support this SOF mode.

*** Marker: DHT (Define Huffman Table) (xFFC4) ***
  OFFSET: 0x00089573
  Huffman table length = 41
  ----
  Destination ID = 1
  Class = 1 (AC Table)
    Codes of length 01 bits (000 total):
    Codes of length 02 bits (003 total): 00 01 11
    Codes of length 03 bits (001 total): 10
    Codes of length 04 bits (000 total):
    Codes of length 05 bits (002 total): 20 21
    Codes of length 06 bits (002 total): 30 31
    Codes of length 07 bits (002 total): 40 50
    Codes of length 08 bits (002 total): 41 60
    Codes of length 09 bits (003 total): 51 70 90
    Codes of length 10 bits (001 total): 61
```

```
    Codes of length 11 bits (001 total): 80
    Codes of length 12 bits (001 total): 71
    Codes of length 13 bits (001 total): A0
    Codes of length 14 bits (001 total): B0
    Codes of length 15 bits (001 total): E0
    Codes of length 16 bits (001 total): C0
    Total number of codes: 022


*** Marker: SOS (Start of Scan) (xFFDA) ***
  OFFSET: 0x0008959E
  Scan header length = 8
  Number of img components = 1
    Component[1]: selector=0x02, table=0(DC),1(AC)
  Spectral selection = 1 .. 63
  Successive approximation = 0x10

  NOTE: Scan parsing doesn't support this SOF mode.

*** Marker: DHT (Define Huffman Table) (xFFC4) ***
  OFFSET: 0x000923B4
  Huffman table length = 46
  ----
  Destination ID = 0
  Class = 1 (AC Table)
    Codes of length 01 bits (000 total):
    Codes of length 02 bits (003 total): 00 01 11
    Codes of length 03 bits (000 total):
    Codes of length 04 bits (002 total): 21 31
    Codes of length 05 bits (002 total): 41 51
    Codes of length 06 bits (002 total): 10 61
    Codes of length 07 bits (002 total): 71 81
    Codes of length 08 bits (002 total): 20 91
    Codes of length 09 bits (002 total): A1 B1
```

```
    Codes of length 10 bits (002 total): 30 C1
    Codes of length 11 bits (002 total): 40 D1
    Codes of length 12 bits (003 total): E1 F0 F1
    Codes of length 13 bits (001 total): 50
    Codes of length 14 bits (001 total): 80
    Codes of length 15 bits (000 total):
    Codes of length 16 bits (003 total): 60 70 90
    Total number of codes: 027


*** Marker: SOS (Start of Scan) (xFFDA) ***
  OFFSET: 0x000923E4
  Scan header length = 8
  Number of img components = 1
    Component[1]: selector=0x01, table=0(DC),0(AC)
  Spectral selection = 1 .. 63
  Successive approximation = 0x10

  NOTE: Scan parsing doesn't support this SOF mode.
```

GIMP optimized Huffman tables:

```
  OFFSET: 0x00000D0B
  Huffman table length = 31
  ----
  Destination ID = 0
  Class = 0 (DC / Lossless Table)
    Codes of length 01 bits (000 total):
    Codes of length 02 bits (000 total):
    Codes of length 03 bits (007 total): 01 02 03 04 05 06 07
    Codes of length 04 bits (001 total): 08
    Codes of length 05 bits (001 total): 00
    Codes of length 06 bits (001 total): 09
    Codes of length 07 bits (001 total): 0A
    Codes of length 08 bits (001 total): 0B
```

```
    Codes of length 09 bits (000 total):
    Codes of length 10 bits (000 total):
    Codes of length 11 bits (000 total):
    Codes of length 12 bits (000 total):
    Codes of length 13 bits (000 total):
    Codes of length 14 bits (000 total):
    Codes of length 15 bits (000 total):
    Codes of length 16 bits (000 total):
    Total number of codes: 012


*** Marker: DHT (Define Huffman Table) (xFFC4) ***
  OFFSET: 0x00000D2C
  Huffman table length = 96
  ----
  Destination ID = 0
  Class = 1 (AC Table)
    Codes of length 01 bits (000 total):
    Codes of length 02 bits (002 total): 01 02
    Codes of length 03 bits (002 total): 03 11
    Codes of length 04 bits (002 total): 04 12
    Codes of length 05 bits (001 total): 21
    Codes of length 06 bits (003 total): 00 05 31
    Codes of length 07 bits (003 total): 06 13 22
    Codes of length 08 bits (003 total): 07 32 41
    Codes of length 09 bits (003 total): 14 23 51
    Codes of length 10 bits (003 total): 08 42 61
    Codes of length 11 bits (003 total): 15 52 71
    Codes of length 12 bits (002 total): 24 33
    Codes of length 13 bits (001 total): 09
    Codes of length 14 bits (002 total): 16 62
    Codes of length 15 bits (000 total):
    Codes of length 16 bits (047 total):
        81 43 72 91 17 34 A1 B1 25 35 53 73 74 82 B2 B3
```

```
        C1 0A 18 44 75 92 B4 C2 D1 E1 F0 26 36 63 A2 C3
        F1 45 19 83 84 54 B5 C4 27 46 76 85 93 D3 E2
    Total number of codes: 077


*** Marker: DHT (Define Huffman Table) (xFFC4) ***
  OFFSET: 0x00000D8E
  Huffman table length = 29
  ----
  Destination ID = 1
  Class = 0 (DC / Lossless Table)
    Codes of length 01 bits (000 total):
    Codes of length 02 bits (002 total): 01 02
    Codes of length 03 bits (003 total): 00 03 04
    Codes of length 04 bits (001 total): 05
    Codes of length 05 bits (001 total): 06
    Codes of length 06 bits (001 total): 07
    Codes of length 07 bits (001 total): 08
    Codes of length 08 bits (001 total): 09
    Codes of length 09 bits (000 total):
    Codes of length 10 bits (000 total):
    Codes of length 11 bits (000 total):
    Codes of length 12 bits (000 total):
    Codes of length 13 bits (000 total):
    Codes of length 14 bits (000 total):
    Codes of length 15 bits (000 total):
    Codes of length 16 bits (000 total):
    Total number of codes: 010


*** Marker: DHT (Define Huffman Table) (xFFC4) ***
  OFFSET: 0x00000DAD
  Huffman table length = 83
  ----
```

```
Destination ID = 1
Class = 1 (AC Table)
  Codes of length 01 bits (001 total): 01
  Codes of length 02 bits (000 total):
  Codes of length 03 bits (002 total): 02 11
  Codes of length 04 bits (001 total): 21
  Codes of length 05 bits (003 total): 12 31 41
  Codes of length 06 bits (003 total): 00 03 51
  Codes of length 07 bits (003 total): 04 22 61
  Codes of length 08 bits (002 total): 32 71
  Codes of length 09 bits (004 total): 05 13 81 91
  Codes of length 10 bits (004 total): 42 A1 B1 F0
  Codes of length 11 bits (005 total): 06 52 C1 D1 E1
  Codes of length 12 bits (003 total): 14 23 62
  Codes of length 13 bits (002 total): F1 72
  Codes of length 14 bits (000 total):
  Codes of length 15 bits (000 total):
  Codes of length 16 bits (031 total):
      07 15 33 82 B2 43 92 A2 16 24 73 C2 B3 53 D2 08
      17 34 63 83 E2 C3 F2 25 44 74 45 54 93 B4 D3
  Total number of codes: 064
```

References

[LD 2004] Ze-Nian Li, Mark S. Drew, *Fundamentals of Multimedia*, Pearson Education International, 2004.

[JO 2015] JPEG.org, `http://www.jpeg.org`.

[Exiv2 2015] `http://dev.exiv2.org/projects/exiv2/wiki/The_Metadata_in_JPEG_files`.

[JM 1999] J. Milano, *Compressed Image File Formats: JPEG, PNG, GIF, XNM, BMP*, ACM Press, 1999.

[FF 2015] FotoForensic.com, `http://fotoforensics.com/tutorial-ela.php`.

[SDC 2015] Stanford Data Compression webpage, `http://cs.stanford.edu/people/eroberts/courses/soco/projects/data-compression/lossy/jpeg/dct.htm`;

[WMC-DCT 2015] Wikimedia Commons Image available at `https://upload.wikimedia.org/wikipedia/commons/2/24/DCT-8x8.png`;

[IA 2015] Impulse Adventure, *JPEG Compression, Quality and File Size*, `http://www.impulseadventure.com/photo/jpeg-compression.html`;

[WMC-ZZ 2015] Wikimedia Commons Image available at `https://en.wikipedia.org/wiki/JPEG#/media/File:JPEG_ZigZag.svg`;

[RY 2015] R. Yerraballi, *Multimedia Systems Concepts Standards and Practice*, `http://users.ece.utexas.edu/~ryerraballi/MSB/pdfs/M4L1.pdf`

[IA-CS 2015] Impulse Adventure, *JPEG chroma subsampling*, `http://www.impulseadventure.com/photo/chroma-subsampling.html`;

[WMC-CS 2015] Wikimedia Commons Image available at `https://upload.wikimedia.org/wikipedia/commons/thumb/5/56/Chroma_subsampling_ratios.svg/2000px-Chroma_subsampling_ratios.svg.png`

[PP 2011] M. Zhang, *A Higher Quality Setting in Photoshop Sometimes Reduces JPEG Quality*, http://petapixel.com/2011/08/26/a-higher-quality-setting-in-photoshop-sometimes-reduces-jpeg-quality/;

[IA-OP 2015] Impulse Adventure, *What is an Optimized JPEG?*, http://www.impulseadventure.com/photo/optimized-jpeg.html;

[GOM 2015] GIMP 2 Online manual, *Chapter 6: Getting images out of GIMP*, http://docs.gimp.org/2.2/en/gimp-images-out.html;