JPEG for the horseshoe crabs β

JPEG compression quickly explained. The JPEG compression algorithm goes trough a few transformation steps namely:

- RGB to YC_bC_r color space transformation
- Chroma subsampling (optional)
- Discrete Cosine Transform (DCT) applied to 8 by 8 matrices
- Quantization of the resulting matrices coefficients
- Lossless encoding (Delta + Huffman coding of AC coefficients, Run Length Encoding of zeroes following a zigzag path + Huffman coding of DC coefficients)

1) RGB to YC_bC_r color space transformation

This color space transformation did not figure in the original JPEG specifications published in 1992, it comes from the JFIF (JPEG File Image Format) de facto standard, but it's now part of ISO/IEC 10918-5 - JPEG Part 5: JPEG File Interchange Format.

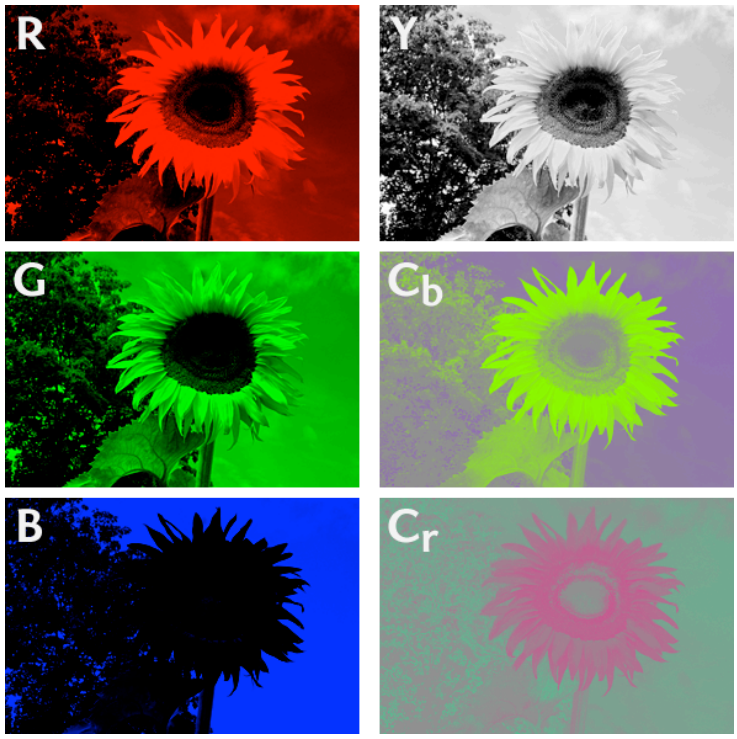
The JPEG compression itself is color agnostic, it simply works on unnamed components (between 1 and 4). Working in the YC_bC_r color space is important since it opens the way to much better compression ratios compared to the RGB color model.

$$
\begin{aligned}
Y &= 0.2990\ R + 0.5870\ G + 0.1140\ B \\
C_b &= -0.1687\ R - 0.3313\ G + 0.5000\ B + 128 \\
C_r &= 0.5000\ R - 0.4187\ G - 0.0813\ B + 128
\end{aligned}
$$

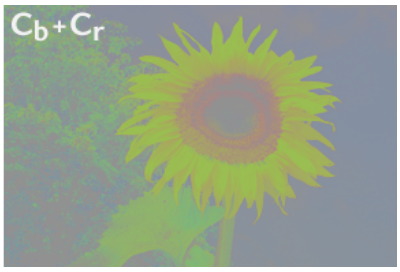
ITU-R BT.601 (formerly CCIR601) as used by JFIF/JPEG

Unfortunately due to rounding errors (values are stored as integers within JPEG) a roundtrip RGB ➜ YC_bC_r ➜ RGB will not reproduce exactly the same colors, JPEG is definitely a lossy image format you have been warned.

JPEG-XR addresses this problem by using a different internal color space, but as its name implies it's also a totally different beast compared to plain old JPEG.



Luminance-chrominance color spaces (YUV, YD_bD_r, YIQ…) pre-date the inception of computer graphics by far since they can be rooted to early works on color television done by Georges Valensi in 1938.

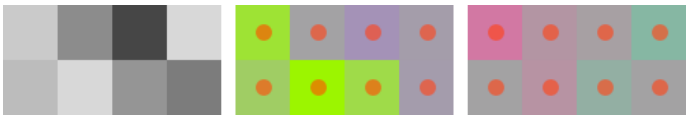


Basically it separates Y —Luma, "grayscale"— from C_bC_r —Chroma, "color"—, since the human visual system is known to be more sensitive to luminance than chrominance (there are way fewer cones than rods in the retina) chrominance is often treated as the poor relation of image components.

2) Chroma subsampling

More bandwidth has always been allocated to luminance than chrominance in the video world, many video encoding schemes (both analog and digital) resort to chroma subsampling to quickly implement this feature. JPEG offers way more subsampling types than those used in video, here are the most common ones expressed using the three part ratio convention.

4:4:4 (1x1) No subsampling



All the chroma samples are kept as is. To ensure maximum picture quality chroma subsampling remains an optional feature within JPEG.

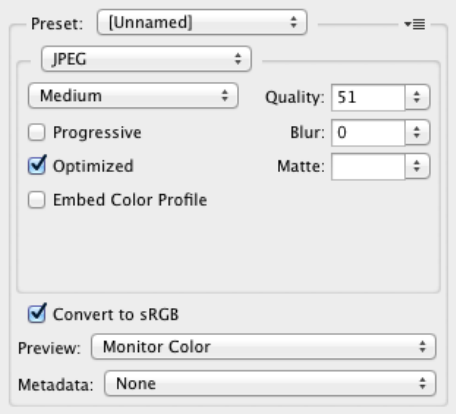
4:2:2 (2x1) Horizontal

Two horizontally contiguous chroma samples are merged into a single one, horizontal chroma definition is thus halved. This type of subsampling is often used by default.

4:2:0 (2x2) Horizontal and vertical

A square of four chroma samples is merged into a single one, chroma definition is divided by four. This is WebP (lossy mode) mandatory subsampling type and common type for highly compressed JPEGs.

Historically the first part of the ratio represented a horizontal sampling reference, and the second two parts represented the number of chrominance samples in the first and second rows of that sampling reference, samples were effectively measured at different frequencies. Since JPEG subsampling is not restricted to only two rows some of its subsampling types cannot be expressed this way.



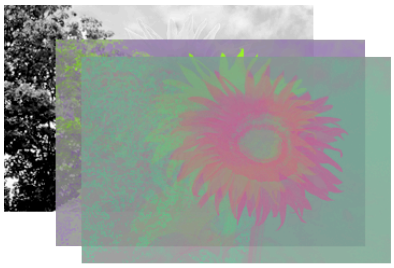
If you are familiar with Adobe Photoshop "Save for Web" dialog you may be puzzled by this chroma subsampling concept since it doesn't appear on this panel.
In fact Photoshop does it in your back based on the sole "Quality" setting, basically it applies no chroma subsampling at all when quality is in the 51-100 range and a rather harsh 4:2:0 subsampling when quality is in the 0-50 range.

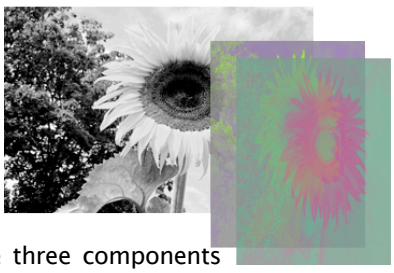
When batch processing files with ImageMagick the option called -sampling-factor gives access to chroma subsampling settings.

Lets compare the visual quality of different chroma subsampling types.

(1x1) 100% of original chroma — 4:4:4

(1x2) 50% of original chroma — 4:4:0





Relative size of the three components

(2x1) 50% of original chroma — 4:2:2

(2x2) 25% of original chroma — 4:2:0

(3x2) 16.6% of original chroma

(2x4) 12.5% of original chroma

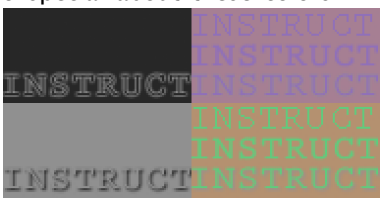
Chroma can even be subsampled down to (4x4) that's 16 times less information. JPEG offers a lot of flexibility at this level for instance C_b and C_r could be subsampled independently i.e. (2x1) and (1x2), since JPEG is color agnostic even luma could be subsampled (this could be used the reduce the file size of zoomed pictures). Fiddling with cjpeg "wizards" settings constitutes the premier venue to unleash all the potential of subsampling.

JPEG is often considered doing a poor job when compressing text areas, color text on a colored background can appear especially blurry and smeared. This is an ill-fated side effect related to chroma subsampling.

Chroma subsampling type

(1x1)	(2x1)	(3x1)
(1x2)	(2x2)	(3x2)
(1x3)	(2x3)	(3x3)

The background and foreground colors used in this sample belong to the worst pairs of colors that can be picked to illustrated this problem. What's so special about these colors?



If you look closely at the luma component you'll notice that there's no trace of the text (the white glow and dark shadow on the third line are attempts to mitigate the blurring effect) this means that practically all the information lies in fact in the chroma components…

Sharp edges and thin lines are difficult to compress since JPEG has been designed to handle more natural photographic pictures in which soft transitions are the norm.

3) Discrete Cosine Transform (DCT)

The DCT is the heart of JPEG compression, this mathematical transform turns a spacial representation (more or less a pixel map) into a spectral representation (pattern frequencies). This describes in fact the forward DCT or FDCT, the reverse transform is called the inverse DCT or IDCT. These operations are heavy-duty tasks, to speed up the (de)compression process the transforms are not applied to the entire image at once but rather on small chunks called data units.

A data unit is an 8 by 8 array, or matrix, holding 64 elements coming straight (they are actually level shifted) from the two dimensional representation of each YC_bC_r component. Since the processing of each such matrix is independent it could be distributed among multiple CPUs or handled by a massively parallel ASIC chip.

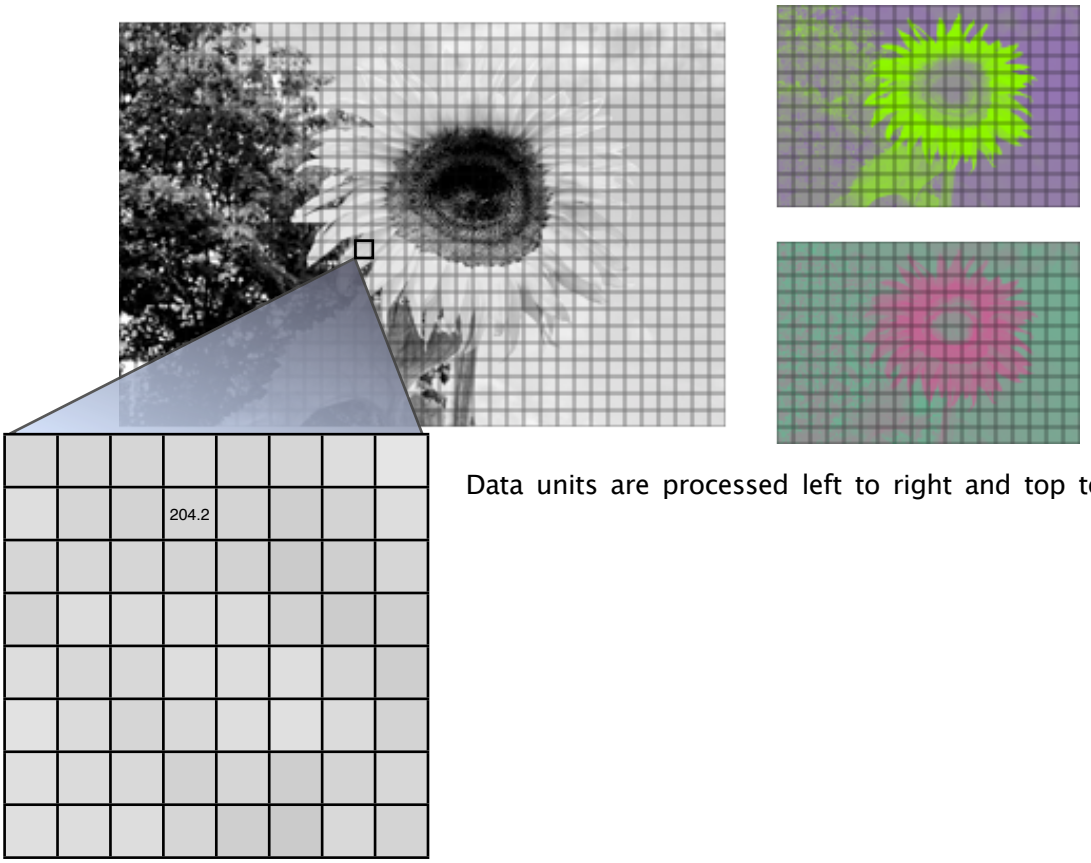
$$\text{FDCT:} \quad S_{vu} = \frac{1}{4} C_u C_v \sum_{x=0}^{7} \sum_{y=0}^{7} s_{yx} \cos \frac{(2x+1)u\pi}{16} \cos \frac{(2y+1)v\pi}{16}$$

$$\text{IDCT:} \quad s_{yx} = \frac{1}{4} \sum_{u=0}^{7} \sum_{v=0}^{7} C_u C_v S_{vu} \cos \frac{(2x+1)u\pi}{16} \cos \frac{(2y+1)v\pi}{16}$$

where

$$C_u, C_v = 1/\sqrt{2} \text{ for } u, v = 0$$

$$C_u, C_v = 1 \text{ otherwise}$$



Data units are processed left to right and top to bottom.