

# JPEG

From Wikipedia, the free encyclopedia

In computing, **JPEG** (pronounced JAY-peg; IPA: [ˈdʒeɪpɛɡ]) is a commonly used standard method of compression for photographic images. The name JPEG stands for Joint Photographic Experts Group, the name of the committee who created the standard. The group was organized in 1986, issuing a standard in 1992 which was approved in 1994 as ISO 10918-1. JPEG should not be confused with MPEG (Moving Picture Experts Group) which produces compression schemes for video.

JPEG itself specifies both the codec, which defines how an image is compressed into a stream of bytes and decompressed back into an image, and the file format used to contain that stream. The compression method is usually lossy compression, meaning that some visual quality is lost in the process, although there are variations on the standard baseline JPEG which are lossless. There is also a "progressive" format, in which data is compressed in multiple passes of progressively higher detail. This is ideal for large images that will be displayed whilst downloading over a slow connection, allowing a reasonable preview before all the data has been retrieved. However, progressive JPEGs are not as widely supported.


The file format is known as 'JPEG Interchange Format', as specified in Annex B of the standard. This is often confused with the JPEG *File* Interchange Format (JFIF), a minimal version of the JPEG format that was deliberately simplified so that it could be widely implemented and thus become the de-facto standard. Most image editing software programs that write to a "JPEG file" are actually creating a file in JFIF format.

Image files that employ JPEG compression are commonly called "JPEG files". The most common file extension for this format is **.jpg**, though `.jpeg`, `.jpe`, `.jfif` and `.jif` are also used. It is also possible for JPEG data to be embedded in other file types, such as TIFF format images.

JPEG/JFIF is the format most used for storing and transmitting photographs on the World Wide Web. For this application, it is preferred to formats such as GIF, which has a limit of 256 distinct colors that is insufficient for color photographs, and PNG, which produces much larger image files for this type of image. The compression algorithm is *not* as well suited for line drawings and other textual or iconic graphics, and thus the PNG and GIF formats are preferred for these types of images.

The MIME media type for JPEG is *image/jpeg* (defined in RFC 1341).

**JPEG**



A photo of a flower compressed with successively more lossy compression ratios from left to right.

<b>File extension:</b>	.jpeg, .jpg, .jpe .jfif, .jfi, .jif (containers)
<b>MIME type:</b>	image/jpeg
<b>Type code:</b>	JPEG
<b>Uniform Type Identifier:</b>	public.jpeg
<b>Developed by:</b>	Joint Photographic Experts Group

## Contents

- 1 JPEG codec
  - 1.1 Encoding
    - 1.1.1 Color space transformation
    - 1.1.2 Downsampling
    - 1.1.3 Block splitting
    - 1.1.4 Discrete cosine transform
    - 1.1.5 Quantization
    - 1.1.6 Entropy coding

- 1.2 Compression ratio and artifacts
- 1.3 Decoding
- 1.4 Required precision
- 1.5 Lossless editing
- 2 JPEG Interchange Format file format
  - 2.1 Shortcomings
    - 2.1.1 Color profile
- 3 Usage
  - 3.1 Photographs
  - 3.2 Medical imaging: JPEG's 12-bit mode
- 4 Potential patent issues
- 5 Standards
- 6 References
- 7 See also
- 8 External links

## JPEG codec

### Encoding

Many of the options in the JPEG standard are not commonly used, and as mentioned above, most image software uses the simpler JFIF format when creating a JPEG file, which amongst other things specifies the encoding method. Here is a brief description of one of the more common methods of encoding when applied to an input that has 24 bits per pixel (eight each of red, green, and blue). This particular option is a lossy data compression method.

### Color space transformation

First, the image should be converted from RGB into a different color space called YCbCr. It has three components Y, Cb and Cr: the Y component represents the brightness of a pixel, the Cb and Cr components represent the chrominance (split into Blue and Red components). This is the same as the color space used by PAL, MAC and digital color television transmission (but not by NTSC, which uses the similar YIQ color space). The YCbCr color space conversion allows greater compression for the same image quality (or greater image quality for the same compression).

This conversion to YCbCr is specified in the JFIF standard, and should be performed for the resulting JPEG file to have maximum compatibility. However, many "high quality" JPEG images do not apply this step and instead keep them in the sRGB color space, where each color plane is compressed and quantized separately with similar quality levels.

### Downsampling

The human eye can see more detail in the Y component (brightness) than in Cb (blue) and Cr (red). Using this knowledge, encoders can be designed to compress images more efficiently.

The above transformation enables the next step, which is to reduce the Cb and Cr components (called "downsampling" or "chroma subsampling"). The ratios at which the downsampling can be done on JPEG are 4:4:4 (no downsampling), 4:2:2 (reduce by factor of 2 in horizontal direction), and most commonly 4:2:0 (reduce by factor of 2 in horizontal and vertical directions). For the rest of the compression process, Y, Cb and Cr are processed separately and in a very similar manner. Downsampling the chroma components saves 33% or 50% of the space taken by the image.

### Block splitting

After subsampling each channel must be split into 8x8 blocks (of pixels), if the data for a channel does not represent an integer number of blocks then the encoder must fill the remaining area of the incomplete blocks with some form of dummy data:

- filling the edge pixels with a fixed color (typically black) creates dark artifacts along the visible part of the border
- repeating the edge pixels is a common but non-optimal technique that avoids the visible border, but it still creates artifacts with the colorimetry of the filled cells
- a better strategy is to fill pixels using colors that preserve the DCT coefficients of the visible pixels, at least for the low frequency ones (for example filling with the average color of the visible part will preserve the first DC coefficient, but best fitting the next two AC coefficients will produce much better results with less visible 8x8 cell edges along the border).

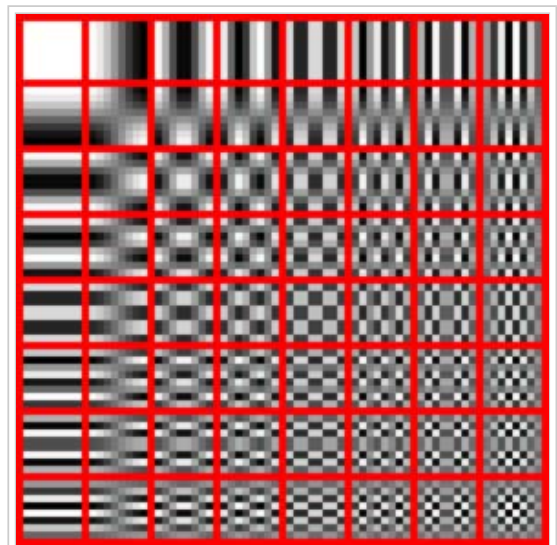
**Discrete cosine transform**

Next, each component (Y, Cb, Cr) of the image is "tiled" into sections of eight by eight pixels each, then each tile is converted to frequency space using a two-dimensional forward discrete cosine transform (DCT, type II).

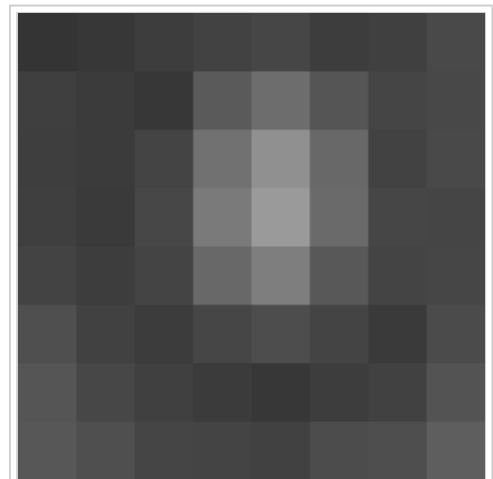
If one such 8x8 8-bit subimage is:

52	55	61	66	70	61	64	73
63	59	55	90	109	85	69	72
62	59	68	113	144	104	66	73
63	58	71	122	154	106	70	69
67	61	68	104	126	88	68	70
79	65	60	70	77	68	58	75
85	71	64	59	55	61	65	83
87	79	69	68	65	76	78	94

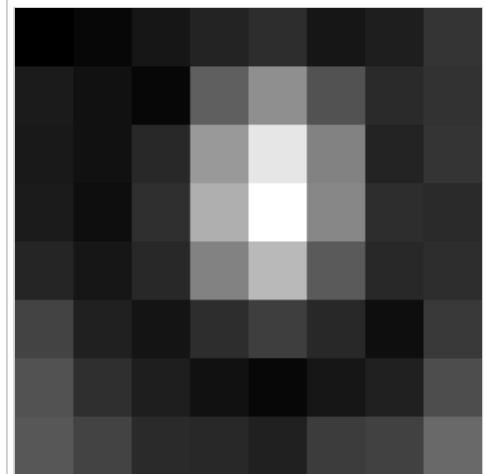
which, if we now subtract 128 from each element, results in



The DCT transforms 64 pixels to a linear combination of these 64 squares



The 8x8 subimage shown in 8-bit greyscale



The 8x8 subimage shown scaled to full 8-bit range greyscale

$$\begin{bmatrix} -76 & -73 & -67 & -62 & -58 & -67 & -64 & -55 \\ -65 & -69 & -73 & -38 & -19 & -43 & -59 & -56 \\ -66 & -69 & -60 & -15 & 16 & -24 & -62 & -55 \\ -65 & -70 & -57 & -6 & 26 & -22 & -58 & -59 \\ -61 & -67 & -60 & -24 & -2 & -40 & -60 & -58 \\ -49 & -63 & -68 & -58 & -51 & -60 & -70 & -53 \\ -43 & -57 & -64 & -69 & -73 & -67 & -63 & -45 \\ -41 & -49 & -59 & -60 & -63 & -52 & -50 & -34 \end{bmatrix}$$

and then taking the DCT and rounding to the nearest integer results in

$$\begin{bmatrix} -415 & -30 & -61 & 27 & 56 & -20 & -2 & 0 \\ 4 & -22 & -61 & 10 & 13 & -7 & -9 & 5 \\ -47 & 7 & 77 & -25 & -29 & 10 & 5 & -6 \\ -49 & 12 & 34 & -15 & -10 & 6 & 2 & 2 \\ 12 & -7 & -13 & -4 & -2 & 2 & -3 & 3 \\ -8 & 3 & 2 & -6 & -2 & 1 & 4 & 2 \\ -1 & 0 & 0 & -2 & -1 & -3 & 4 & -1 \\ 0 & 0 & -1 & -4 & -1 & 0 & 1 & 2 \end{bmatrix}$$



The compressed 8x8-squares are visible in the scaled up picture, together with other visual artifacts of the lossy compression

Note the rather large value of the top-left corner. This is the DC coefficient. The remaining 63 coefficients are called the AC coefficients. The discrete cosine transform actually temporarily increases the size of the image, since the DCT coefficients of a 8-bit/component image take up to 11 or 12 bits (depending on fidelity of the DCT calculation) to store. This may force the codec to temporarily use 16-bit bins to hold these coefficients doubling the formal size of the image representation at this point. The advantage of the discrete cosine transform is its tendency to aggregate most of the signal in one corner of the result, as may be seen above. The quantization step to follow accentuates this effect while simultaneously reducing the size of the DCT coefficients to 8 bits or less, resulting in a signal with a large trailing region containing zeros that the entropy stage can simply throw away. The temporary increase in size at this stage is not a performance concern for most JPEG implementations, because typically only a very small part of the image is stored in full DCT form at any given time during the encoding or decoding process.

### Quantization

The human eye is good at seeing small differences in brightness over a relatively large area, but not so good at distinguishing the exact strength of a high frequency brightness variation. This fact allows one to get away with greatly reducing the amount of information in the high frequency components. This is done by simply dividing each component in the frequency domain by a constant for that component, and then rounding to the nearest integer. This is the main lossy operation in the whole process. As a result of this, it is typically the case that many of the higher frequency components are rounded to zero, and many of the rest become small positive or negative numbers, which take many fewer bits to store.

A common quantization matrix is:

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

Using this quantization matrix with the DCT coefficient matrix from above results in:

$$\begin{bmatrix} -26 & -3 & -6 & 2 & 2 & -1 & 0 & 0 \\ 0 & -2 & -4 & 1 & 1 & 0 & 0 & 0 \\ -3 & 1 & 5 & -1 & -1 & 0 & 0 & 0 \\ -4 & 1 & 2 & -1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

For example, using  $-415$  (the DC coefficient) and rounding to the nearest integer

$$\text{round}\left(\frac{-415}{16}\right) = \text{round}(-25.9375) = -26$$

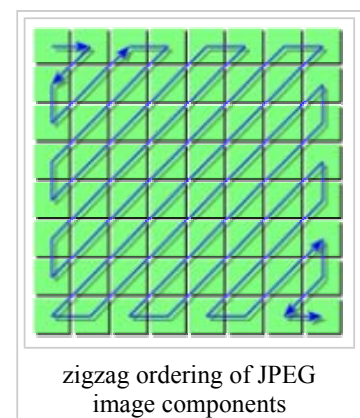
### Entropy coding

Entropy coding is a special form of lossless data compression. It involves arranging the image components in a "zigzag" order employing run-length encoding (RLE) algorithm that groups similar frequencies together, inserting length coding zeros, and then using Huffman coding on what is left.

The JPEG standard also allows, but does not require, the use of arithmetic coding which is mathematically superior to Huffman coding. However, this feature is rarely used as it is covered by patents and because it is much slower to encode and decode compared to Huffman coding. Arithmetic coding typically makes files about 5% smaller.

The zig-zag sequence for the above quantized coefficients would be:

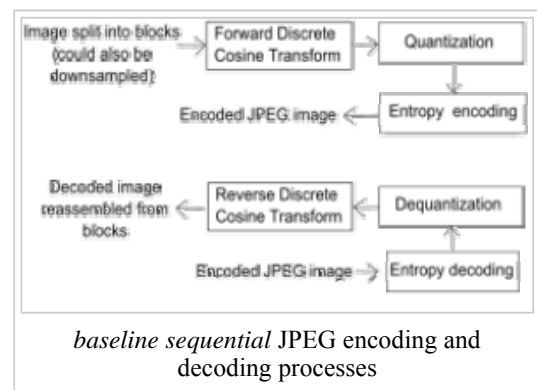
$-26,$



-3, 0,  
 -3, -2, -6,  
 2, -4, 1, -4,  
 1, 1, 5, 1, 2,  
 -1, 1, -1, 2, 0, 0,  
 0, 0, 0, -1, -1, 0, 0,  
 0, 0, 0, 0, 0, 0, 0, 0,  
 0, 0, 0, 0, 0, 0, 0,  
 0, 0, 0, 0, 0, 0,  
 0, 0, 0, 0, 0,  
 0, 0, 0, 0,  
 0, 0, 0,  
 0, 0,  
 0

If the *i*-th block is represented by  $B_i$  and positions within each block are represented by  $(p,q)$  where  $p = 0, 1, \dots, 7$  and  $q = 0, 1, \dots, 7$ , then any co-efficient in the DCT image can be represented as  $B_i(p,q)$ . Thus, in the above scheme, the order of encoding pixels (for the *i*-th block) is  $B_i(0,0)$ ,  $B_i(0,1)$ ,  $B_i(1,0)$ ,  $B_i(2,0)$ ,  $B_i(1,1)$ ,  $B_i(0,2)$ ,  $B_i(0,3)$ ,  $B_i(1,2)$  and so on.

This encoding mode is called baseline *sequential* encoding. Baseline JPEG also supports *progressive* encoding. While sequential encoding encodes co-efficients of a single block at a time (in a zig-zag manner), progressive encoding encodes similar-positioned coefficients of all blocks in one go, followed by the next positioned co-efficients of all blocks, and so on. So, if the image is divided into  $N$   $8 \times 8$  blocks  $\{B_0, B_1, B_2, \dots, B_{N-1}\}$ , then progressive encoding encodes  $B_i(0,0)$  for all blocks, i.e. for all  $i = 0, 1, 2, \dots, N-1$ . This is followed by encoding  $B_i(0,1)$  co-efficient of all blocks, followed by  $B_i(1,0)$ -th co-efficient of all blocks, then  $B_i(0,2)$ -th co-efficient of all blocks, and so on. It should be noted here that once all similar-positioned co-efficients have been encoded, the next position to be encoded is the one occurring next in the zig-zag traversal as indicated in the figure above. It has been found that Baseline Progressive JPEG encoding usually gives better compression as compared to Baseline Sequential JPEG though the difference is not too large.



In the rest of the article, it is assumed that the co-efficient pattern generated is due to sequential mode.

In order to encode the above generate co-efficient pattern, JPEG uses Huffman encoding. JPEG has a special Huffman code word for ending the sequence prematurely when the remaining coefficients are zero.

Using this special code word: "EOB", the sequence becomes:

-26,  
 -3, 0,  
 -3, -2, -6,  
 2, -4, 1, -4,  
 1, 1, 5, 1, 2,  
 -1, 1, -1, 2, 0, 0,  
 0, 0, 0, -1, -1, EOB

JPEG's other code words represent combinations of (a) the number of significant bits of a coefficient, including sign, and (b) the number of consecutive zero coefficients that follow it. (Once you know how many bits to expect, it takes 1 bit to represent the choices  $\{-1, +1\}$ , 2 bits to represent the choices  $\{-3, -2, +2, +3\}$ , and so forth.) In our example block, most of the quantized coefficients are small numbers that are not followed immediately by a zero coefficient. These more-frequent cases will be represented by shorter code words.

The JPEG standard provides general-purpose huffman tables; encoders may also choose to generate huffman tables optimized for the actual frequency distributions in images being encoded.

### Compression ratio and artifacts

The resulting compression ratio can be varied according to need by being more or less aggressive in the divisors used in the quantization phase. Ten to one compression usually results in an image that cannot be distinguished by eye from the original. 100 to one compression is usually possible, but will look distinctly artifacted compared to the original. The appropriate level of compression depends on the use to which the image will be put.

Those who use the World Wide Web may be familiar with the irregularities known as compression artifacts that appear in JPEG images. These are due to the quantization step of the JPEG algorithm. They are especially noticeable around eyes in pictures of faces. They can be reduced by choosing a lower level of compression; they may be eliminated by saving an image using a lossless file format, though for photographic images this will usually result in a larger file size. Compression artifacts make low-quality JPEGs unacceptable for storing heightmaps. The images created with ray-tracing programs have noticeable blocky shapes on the terrain.

Some programs allow the user to vary the amount by which individual blocks are compressed. Stronger compression is applied to areas of the image that show fewer artifacts. This way it is possible to make a JPEG file smaller and prettier by hand.

### Decoding

Decoding to display the image consists of doing all the above in reverse.

Taking the DCT coefficient matrix (after adding the difference of the DC coefficient back in)

$$\begin{bmatrix} -26 & -3 & -6 & 2 & 2 & -1 & 0 & 0 \\ 0 & -2 & -4 & 1 & 1 & 0 & 0 & 0 \\ -3 & 1 & 5 & -1 & -1 & 0 & 0 & 0 \\ -4 & 1 & 2 & -1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

and multiplying it by the quantization matrix from above results in



This image shows the (accentuated) difference between an image saved as JPEG and the original. Note especially the changes occurring near sharp edges.



$$\begin{bmatrix} -416 & -33 & -60 & 32 & 48 & -40 & 0 & 0 \\ 0 & -24 & -56 & 19 & 26 & 0 & 0 & 0 \\ -42 & 13 & 80 & -24 & -40 & 0 & 0 & 0 \\ -56 & 17 & 44 & -29 & 0 & 0 & 0 & 0 \\ 18 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

which closely resembles the original DCT coefficient matrix for the top-left portion. Taking the inverse DCT (type-III DCT) results in an image with values (still shifted down by 128)

$$\begin{bmatrix} -68 & -65 & -73 & -70 & -58 & -67 & -70 & -48 \\ -70 & -72 & -72 & -45 & -20 & -40 & -65 & -57 \\ -68 & -76 & -66 & -15 & 22 & -12 & -58 & -61 \\ -62 & -72 & -60 & -6 & 28 & -12 & -59 & -56 \\ -59 & -66 & -63 & -28 & -8 & -42 & -69 & -52 \\ -60 & -60 & -67 & -60 & -50 & -68 & -75 & -50 \\ -54 & -46 & -61 & -74 & -65 & -64 & -63 & -45 \\ -45 & -32 & -51 & -72 & -58 & -45 & -45 & -39 \end{bmatrix}$$

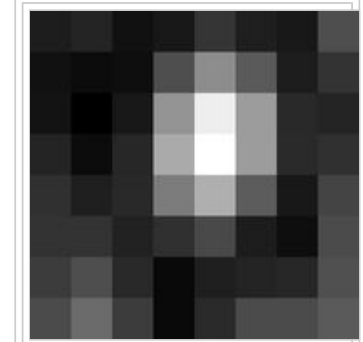
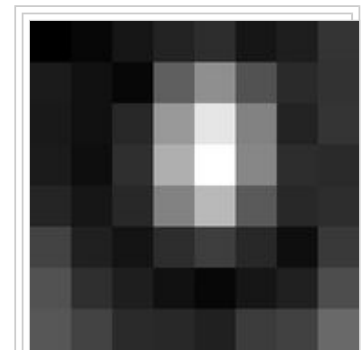
and adding 128 to each entry

$$\begin{bmatrix} 60 & 63 & 55 & 58 & 70 & 61 & 58 & 80 \\ 58 & 56 & 56 & 83 & 108 & 88 & 63 & 71 \\ 60 & 52 & 62 & 113 & 150 & 116 & 70 & 67 \\ 66 & 56 & 68 & 122 & 156 & 116 & 69 & 72 \\ 69 & 62 & 65 & 100 & 120 & 86 & 59 & 76 \\ 68 & 68 & 61 & 68 & 78 & 60 & 53 & 78 \\ 74 & 82 & 67 & 54 & 63 & 64 & 65 & 83 \\ 83 & 96 & 77 & 56 & 70 & 83 & 83 & 89 \end{bmatrix}$$

This is the uncompressed subimage and can be compared to the original subimage (also see images to the right) by taking the difference (original - uncompressed) results in error values

$$\begin{bmatrix} -8 & -8 & 6 & 8 & 0 & 0 & 6 & -7 \\ 5 & 3 & -1 & 7 & 1 & -3 & 6 & 1 \\ 2 & 7 & 6 & 0 & -6 & -12 & -4 & 6 \\ -3 & 2 & 3 & 0 & -2 & -10 & 1 & -3 \\ -2 & -1 & 3 & 4 & 6 & 2 & 9 & -6 \\ 11 & -3 & -1 & 2 & -1 & 8 & 5 & -3 \\ 11 & -11 & -3 & 5 & -8 & -3 & 0 & 0 \\ 4 & -17 & -8 & 12 & -5 & -7 & -5 & 5 \end{bmatrix}$$

with an average absolute error of about 5 values per pixels (i.e.,  $\frac{1}{64} \sum_{x=1}^8 \sum_{y=1}^8 |e(x, y)| = 4.8125$ ).



Notice the slight differences between the original (top) and decompressed image (bottom) which is most readily seen in the bottom-left corner.



The error is most noticeable in the bottom-left corner where the bottom-left pixel becomes darker than the pixel to its immediate right.

### Required precision

The JPEG encoding does not fix the precision needed for the output compressed image. On the contrary, the JPEG standard (as well as the derived MPEG standards) have very strict precision requirements for the decoding, including all parts of the decoding process (variable length decoding, inverse DCT, dequantization, renormalization of outputs) ; the output from the reference algorithm must not exceed :

- a maximum 1 bit of difference for each pixel component
- low mean square error over each 8x8-pixel block
- very low mean error over each 8x8-pixel block
- very low mean square error over the whole image
- extremely low mean error over the whole image

These assertions are tested on a large set of randomized input images, to handle the worst cases. Look at the IEEE 1880-1990 standard for reference. This has a consequence on the implementation of decoders, and it is extremely critical because some encoding processes (notably used for encoding sequences of images like MPEG need to be able to construct, on the encoder side, a reference decoded image). In order to support 8-bit precision per pixel component output, dequantization and inverse DCT transforms are typically implemented with at least 14-bit precision in optimized decoders.

### Lossless editing

A number of alterations can be performed to a JPEG image without any quality loss. Blocks can be rotated in 90 degree increments, flipped in the horizontal vertical and diagonal axes and moved about in the image. Not all blocks from the original image need to be used in the modified one.

The top and left of a JPEG image must lie on a block boundary, but the bottom and right need not do so. This limits the possible lossless crop operations, and also what flips and rotates can be performed on an image whose edges do not lie on a block boundary for all channels.

It is also possible to transform between baseline and progressive formats without any loss of quality, since the only difference is the order in which the coefficients are placed in the file.

Furthermore, if a JPEG is decompressed, edited, and then recompressed using the same chroma subsampling and quantisation tables, then unedited areas of the image should have minimal quality loss.

When using lossless cropping, it is important to realise that if the bottom or left side of the crop region is not on a block boundary then the rest of the data from the partially used blocks will still be present in the cropped file and can be recovered relatively easily by anyone with a hex editor and an understanding of the format.

## JPEG Interchange Format file format

### Shortcomings

The JPEG Interchange Format suffers from three shortcomings which prevent it from being easily understood:

- Color Space definition
- Component Sub-Sampling Registration definition
- Pixel Aspect Ratio definition

Several further standards have attempted to rectify this situation. These include, 'JPEG File Interchange Format' (JFIF), 'Exchangeable image file format' (Exif) and ICC color profiles.

## Color profile

Many JPEG files embed an ICC color profile (color space). Commonly used color profiles include sRGB and Adobe RGB. Because these color spaces use a non-linear transformation, the dynamic range of an 8-bit JPEG file is about 11 stops.

## Usage

JPEG is at its best on photographs and paintings of realistic scenes with smooth variations of tone and color. In this case it usually performs much better than purely lossless methods while still giving a good looking image. In fact, it will usually produce much better results for such images than, for example, GIF, which can be lossless as long as the image contains 256 or fewer unique colors but requires severe quantization for full-color images.

Some operations on JPEG images, such as rotation by multiples of 90°, can be performed losslessly as long as the image size is a multiple of eight pixels in both directions. One program which can do this is the `jpegtran` utility which comes with the reference implementation.

## Photographs

JPEG compression artifacts blend well into photographs with detailed non-uniform textures, allowing higher compression ratios. Notice how a higher compression ratio first affects the high-frequency textures in the upper-left corner of the image, and how the contrasting lines become more fuzzy. The very high compression ratio severely affects the quality of the image, although the overall colors and image form is still recognizable. However, the precision of colors suffer less (for a human eye) than the precision of contours (based on luminance). This justifies the fact that images should be first transformed in a color model separating the luminance from the chromatic information, before subsampling the chromatic planes (which may also use lower quality quantization) in order to preserve the precision of the luminance plane with more information bits.

For information, the uncompressed 24-bit RGB bitmap image below (73,242 pixels) would require 219,726 bytes (excluding all other information headers). The filesizes indicated below include the internal JPEG information headers and some meta-data. For full quality images (Q=100), about 8.25 bits per color pixel is required. On grayscale images, a minimum of 6.5 bits per pixel is enough (a comparable Q=100 quality color information requires about 25% more encoded bits). The full quality image below (Q=100) is encoded at 9 bits per color pixel, the medium quality image (Q=25) uses 1 bit per color pixel. For most applications, the quality factor should not go below 0.75 bit per pixel (Q=12.5), as demonstrated by the low quality image. The image at lowest quality uses only 0.13 bit per pixel, and displays very poor color, it could only be usable after subsampling to a much lower display size.



Full quality (Q = 100), filesize 83,261 B.



Average quality (Q = 50), filesize 15,138 B.



Medium quality (Q = 25), filesize 9,553 B.



Low quality (Q = 10), filesize 4,787 B.



Lowest quality (Q = 1), filesize 1,523 B.

NOTE: The above images are not IEEE / CCIR / EBU test images, and the encoder settings are not specified or available.

The mid-quality photo uses only one sixth the storage space but has little noticeable loss of detail or visible artifacts. However, once a certain threshold of compression is passed, compressed images show increasingly visible defects. See the article on rate distortion theory for a mathematical explanation of this threshold effect.

### Medical imaging: JPEG's 12-bit mode

There are many medical imaging systems that create and process 12-bit JPEG images. The 12-bit JPEG format has been part of the JPEG specification for some time, but very few consumer programs (including web browsers) support this rarely used JPEG format.

### Potential patent issues

In 2002 Forgent Networks asserted that it owned and would enforce patent rights on the JPEG technology, arising from a patent that had been filed on October 27, 1986, and granted on October 6, 1987 (U.S. Patent 4,698,672). The announcement created a furor reminiscent of Unisys' attempts to assert its rights over the GIF image compression standard.

The JPEG committee investigated the patent claims in 2002 and were of the opinion that they were invalidated by prior art.<sup>[1]</sup> Others also concluded that Forgent did not have a patent that covered JPEG.<sup>[2]</sup> Nevertheless, between 2002 and 2004 Forgent was able to obtain about US\$90 million by licensing their patent to some 30 companies. In April 2004, Forgent sued 31 other companies to enforce further license payments. In July of the same year, a consortium of 21 large computer companies filed a countersuit, with the goal of invalidating the

patent. Surprisingly, in contrast to the other major computer companies such as Sony and Philips, Microsoft launched a major lawsuit against Forgent. In February 2006, the United States Patent and Trademark Office agreed to re-examine Forgent's JPEG patent at the request of the Public Patent Foundation.<sup>[3]</sup> On May 26, 2006 the USPTO found the patent invalid based on prior art. The USPTO also found that Forgent knew about the prior art, and did not tell the Patent Office, making any appeal to reinstate the patent highly unlikely to succeed.<sup>[4]</sup>

Forgent also possesses a similar patent granted by the European Patent Office in 1994, though it is unclear how enforceable it is.<sup>[5]</sup>

As of October 6, 2006, the U.S. patent's 20-year term appears to have expired, and in November 2006, Forgent agreed to abandon enforcement of patent claims against use of the JPEG standard.<sup>[6]</sup>

The JPEG committee has as one of its explicit goals that their standards (in particular their baseline methods) be implementable without payment of license fees, and they have secured appropriate license rights for their upcoming JPEG 2000 standard from over 20 large organizations.

## Standards

- JPEG (lossy and lossless): ITU-T T.81, ISO/IEC IS 10918-1
- JPEG (extensions): ITU-T T.84
- JPEG-LS (lossless, improved): ITU-T T.87, ISO/IEC IS 14495-1
- JBIG (black and white pictures): ITU-T T.82, ISO/IEC IS 11544-1
- JPEG 2000 (successor of JPEG/JPEG-LS): ITU-T T.800, ISO/IEC IS 15444-1
- JPEG-2000 (extensions): ITU-T T.801

## References

1. ^ Concerning recent patent claims
2. ^ JPEG and JPEG2000 - Between Patent Quarrel and Change of Technology
3. ^ Trademark Office Re-examines Forgent JPEG Patent
4. ^ USPTO: Broadest Claims Forgent Asserts Against JPEG Standard Invalid
5. ^ Coding System for Reducing Redundancy
6. ^ JPEG Patent Claim Surrendered. Published November 2, 2006; retrieved November 3, 2006.

## See also

- Image compression
- Image file formats
- Comparison of graphics file formats
- Windows Picture and Fax Viewer
- Exchangeable image file format (EXIF)
- JPEG File Interchange Format (JFIF)
- Design rule for Camera File system (DCF)
- JPEG 2000
- Motion JPEG
- Graphics editing program
- GDI+ vulnerability section of GDI article, exploitable bug in JPEG handling code of GDI+ library
- Comparison of layout engines (graphics)
- Generation loss
- PNG
- Lossless Image Codec FELICS
- Libjpeg of Independent JPEG Group
- Deblocking filter (video), the similar deblocking methods could be applied to JPEG

## External links

- JPEG Standard (JPEG ISO/IEC 10918-1 ITU-T Recommendation T.81) or as HTML

- Official Joint Photographic Experts Group site
- JPEG FAQ
- Wotsit.org's entry on the JPEG format
- JFIF File Format as PDF or HTML
- The JPEG Still Picture Compression Standard, Summary by Gregory K. Wallace (Gzipped PostScript file)
- JPEG Compression (Gernot Hoffman)
- Article about hidden data in JPEG files
- More about hidden extras, plus a program to remove them
- Open list of JPEG resources
- DCTlab: Matlab GUI
- Jim M. Goldstein: RAW vs JPEG: Is Shooting RAW Format For Me?
- Oskar Breuning: JPEG Compression: Data Loss & Image Impact
- Comparison of 9 modern JPEG-2000 codecs with JPEG

<b>Multimedia compression formats</b>				[hide]
<b>Video compression formats</b>	<b>ISO/IEC</b>	<b>ITU-T</b>	<b>Others</b>	
	MPEG-1 · MPEG-2 · MPEG-4 ASP · MPEG-4/AVC	H.261 · H.262 · H.263 · H.264	AVS · Bink · Dirac · Indeo · MJPEG · RealVideo · Theora · VC-1 · VP6 · VP7 · WMV	
<b>Audio compression formats</b>	<b>ISO/IEC MPEG</b>	<b>ITU-T</b>	<b>Others</b>	
	MPEG-1 Layer III (MP3) · MPEG-1 Layer II · AAC · HE- AAC	G.711 · G.722 · G.722.1 · G.722.2 · G.723 · G.723.1 · G.726 · G.728 · G.729 · G.729.1 · G.729a	AC3 · Apple Lossless · ATRAC · FLAC · iLBC · Monkey's Audio · μ-law · Musepack · Nellymoser · RealAudio · SHN · Speex · Vorbis · WavPack · WMA · TAK	
<b>Image compression formats</b>	<b>ISO/IEC/ITU-T</b>		<b>Others</b>	
	JPEG · JPEG 2000 · lossless JPEG · JBIG · JBIG2 · PNG · WBMP		APNG · ICER · MNG · BMP · GIF · ILBM · PCX · TGA · TIFF · HD Photo	
<b>Media container formats</b>	<b>General</b>			<b>Audio only</b>
	3GP · ASF · AVI · DMF · DPX · FLV · Matroska · MP4 · MXF · NUT · Ogg · Ogg Media · QuickTime · RealMedia · VOB			AIFF · AU · WAV

Retrieved from "http://en.wikipedia.org/wiki/JPEG"

Categories: Articles to be expanded since January 2007 | All articles to be expanded | Graphics file formats | ISO standards | Lossy compression algorithms

- This page was last modified 14:17, 14 May 2007.
- All text is available under the terms of the GNU Free Documentation License. (See **Copyrights** for details.)  
Wikipedia® is a registered trademark of the Wikimedia Foundation, Inc., a US-registered 501(c)(3) tax-deductible nonprofit charity.