General JPEG Questions & Answers

[Note that these questions and answers are taken almost verbatim from the text of the JPEG Frequently Asked Questions list, maintained by Tom Lane. A lot of the information contained here does not pertain very specifically to JPEGView, but rather to the current state of JPEG standards and to the Independent JPEG Group's software. The latter consists of two general conversion programs: cjpeg, which converts files to JPEG; and djpeg, which converts files from JPEG. Finally, note that references to compression quality factors in this section refer to cjpeg and djpeg, whose quality varies from 0 to 100, rather than from 0.00 to 4.00, as with Apple's QuickTime. For more information about JPEG in general or the free JPEG software in particular, contact the Independent JPEG Group at jpeg-info@uunet.uu.net.]

• What is JPEG?

JPEG (pronounced "jay-peg") is a standardized image compression mechanism. JPEG stands for Joint Photographic Experts Group, the original name of the committee that wrote the standard. JPEG is designed for compressing either full-color or gray-scale digital images of "natural", real-world scenes. It does not work so well on non-realistic images, such as cartoons or line drawings.

JPEG is "lossy", meaning that the image you get out of decompression isn't quite identical to what you originally put in. The algorithm achieves much of its compression by exploiting known limitations of the human eye, notably the fact that small color details aren't perceived as well as small details of light-and-dark. Thus, JPEG is intended for compressing images that will be looked at by humans. If you plan to machine-analyze your images, the small errors introduced by JPEG may be a problem for you, even if they are invisible to the eye.

A useful property of JPEG is that the degree of lossiness can be varied by adjusting compression parameters. This means that the image maker can trade off file size against output image quality. You can make extremely small files if you don't mind poor quality; this is useful for indexing image archives, making thumbnail views or icons, etc., etc. Conversely, if you aren't happy with the output quality at the default compression setting, you can jack up the quality until you are satisfied, and accept lesser compression.

• Why use JPEG?

There are two good reasons: to make your image files smaller, and to store 24-bit-per-pixel color data instead of 8-bit-per-pixel data.

Making image files smaller is a big win for transmitting files across networks and for archiving libraries of images. Being able to compress a 2 Mbyte full-color file down to 100 Kbytes or so makes a big difference in disk space and transmission time! (If you are comparing GIF and JPEG, the size ratio is more like four to one. More details below.)

Unless your viewing software supports JPEG directly, you'll have to convert JPEG to some other format for viewing or manipulating images. Even with a JPEG-capable viewer, it takes longer to decode and view a JPEG image than to view an image of a simpler format (GIF, for instance). Thus, using JPEG is essentially a time/space tradeoff: you give up some time in order to store or transmit an image more cheaply.

It's worth noting that when network or phone transmission is involved, the time savings from transferring a shorter file can be much greater than the extra time to decompress the file. I'll let you do the arithmetic yourself.

The other reason why JPEG will gradually replace GIF as the standard image exchange format is that JPEG can store full color information: 24 bits/pixel (16 million colors) instead of 8 or less (256 or fewer colors). If you have only 8-bit display hardware then this may not seem like much of an advantage to you. Within a couple of years, though, 8-bit GIF will look as obsolete as black-and-white MacPaint format does today. Furthermore, for reasons below, JPEG is far more useful than GIF for exchanging images among people with widely varying color display hardware. Hence JPEG is considerably more appropriate than GIF for use as an image exchange standard.

• When should I use JPEG, and when should I stick with GIF?

As a rule of thumb, JPEG is superior to GIF for storing full-color or gray-scale images of "realistic" scenes; that means scanned photographs and similar material. JPEG is superior even if you don't have 24-bit display hardware, and it is a lot superior if you do.

GIF does significantly better on images with only a few distinct colors, such as cartoons and line drawings. In particular, large areas of pixels that are all exactly the same color are compressed very efficiently indeed by GIF. JPEG can't squeeze these files as much as GIF does without introducing visible defects. This sort of image is best kept in GIF form.

JPEG also has a hard time with very sharp edges: a row of pure-black pixels adjacent to a row of pure-white pixels, for example. Sharp edges tend to come out blurred unless you use a very high quality setting. Again, this sort of thing is not found in scanned photographs, but it shows up fairly often in GIF files: borders, overlaid text, etc. The blurriness is particularly objectionable with text that's only a few pixels high. If you have a GIF with a lot of small-size overlaid text, don't JPEG it.

Computer-drawn images (ray-traced scenes, for instance) usually fall between scanned images and cartoons in terms of complexity. The more complex and subtly rendered the image, the more likely that JPEG will do well on it.

Plain black-and-white (two level) images should never be converted to JPEG. You need at least about 16 gray levels before JPEG is useful for gray-scale images.

• What's all this hoopla about color quantization?

Most people don't have full-color (24 bit per pixel) display hardware. Typical display hardware stores 8 or fewer bits per pixel, so it can display 256 or fewer distinct colors at a time. To display a full-color image, the computer must map the image into an appropriate set of representative colors. This process is called "color quantization".

Clearly, color quantization is a lossy process. It turns out that for most images, the details of the color quantization algorithm have much more impact on the final image quality than do any errors introduced by JPEG (except at the very lowest JPEG quality settings).

Since JPEG is a full-color format, converting a color JPEG image for display on 8-bit-or-less hardware requires color quantization. This is true for all color JPEGs: even if you feed a 256-or-less-color GIF into JPEG, what comes out of the decompressor is not 256 colors, but thousands of colors. JPEG's lossiness affects each pixel a little differently, so two pixels that started with identical colors will probably come out with slightly different colors. Each original color gets "smeared" into a group of nearby colors. Therefore quantization is always required to display a color JPEG on a colormapped display, regardless of the image source. The only way to avoid quantization is to ask for gray-scale output.

On the other hand, a GIF image by definition has already been quantized to 256 or fewer colors. For purposes of picture distribution, GIF has the advantage that the creator precomputes the color quantization, so the end user doesn't have to. This is also the disadvantage of GIF: you're stuck with the creator's quantization. If the creator quantized to a different number of colors than what you can display, you have to re-quantize, resulting in much poorer image quality than if you had quantized once from a full-color image. Furthermore, if the creator didn't use a high-quality color quantization algorithm, you're out of luck.

For this reason, JPEG offers the promise of significantly better image quality for all users whose machines don't match the creator's display hardware. JPEG's full color image can be quantized to precisely match the user's display hardware. Furthermore, you will be able to take advantage of future improvements in quantization algorithms (there is a lot of active research in this area), or purchase better display hardware, to get a better view of JPEG images you already have. With a GIF, you're stuck forevermore with the original quantization.

Finally, an ever-growing number of people have better-than-8-bit display hardware already; even a low-end LC II can give you 16-bits-per-pixel on a standard 12" color monitor. For these people, GIF is already obsolete, as it cannot represent an image to the full capabilities of their display. JPEG images can drive these displays much more effectively. Thus, JPEG is an all-around better choice than GIF for representing images in a machine-independent fashion.

• Why all the argument about file formats?

Strictly speaking, JPEG refers only to a family of compression algorithms; it does not refer to a specific image file format. The JPEG committee was prevented from defining a file format by turf wars within the international standards organizations.

Since we can't actually exchange images with anyone else unless we agree on a common file format, this leaves us with a problem. In the absence of official standards, a number of JPEG program writers have just gone off to "do their own thing", and as a result their programs aren't compatible with anybody else's.

The closest thing we have to a de-facto standard JPEG format is some work that's been coordinated by people at C-Cube Microsystems. They have defined two JPEG-based file formats:

• JFIF (JPEG File Interchange Format), a "low-end" format that transports pixels and not much else.
• TIFF/JPEG, aka TIFF 6.0, an extension of the Aldus TIFF format. TIFF is a "high-end" format that will let you record just about everything you ever wanted to know about an image, and a lot more besides :-). TIFF is a lot more complex than JFIF, and may well prove less transportable, because different vendors have historically implemented slightly different and incompatible subsets of TIFF. It's not likely that adding JPEG to the mix will do anything to improve this situation.

Both of these formats were developed with input from all the major vendors of JPEG-related products; it's reasonably likely that future commercial products will adhere to one or both standards.

I believe that Usenet should adopt JFIF as the replacement for GIF in picture postings. JFIF is simpler than TIFF and is available now; the TIFF 6.0 spec has only recently been officially adopted, and it is still unusably vague on some crucial details. Even when TIFF/JPEG is well defined, the JFIF format is likely to be a widely supported "lowest common denominator"; TIFF/JPEG files may never be as transportable.

A particular case that people may be interested in is Apple's QuickTime software for the Macintosh. QuickTime uses a JFIF-compatible format wrapped inside the Mac-specific PICT structure. Conversion between JFIF and QuickTime JPEG is pretty straightforward, and several Mac programs are available to do it.

Another particular case is Handmade Software's shareware JPEG programs (GIF2JPG/JPG2GIF for MS-DOS, Image Alchemy for MS-DOS and a few Unix platforms). These programs are capable of reading and writing JFIF format. By default, though, they write a proprietary format developed by HSI. This format is NOT readable by any non-HSI programs and should not be
used for Usenet postings. Use the -j switch to get JFIF output.

• Does loss accumulate with repeated compression/decompression?

It would be nice if, having compressed an image with JPEG, you could decompress it, manipulate it (crop off a border, say), and recompress it without any further image degradation beyond what you lost initially. Unfortunately this is not the case. In general, recompressing an altered image loses more information, though usually not as much as was lost the first time around.

The next best thing would be that if you decompress an image and recompress it without changing it then there is no further loss, i.e., you get an identical JPEG file. Even this is not true; at least, not with the current free JPEG software. It's essentially a problem of accumulation of roundoff error. If you repeatedly compress and decompress, the image will eventually degrade to where you can see visible changes from the first-generation output. (It usually takes many such cycles to get visible change.) One of the things on our to-do list is to see if accumulation of error can be avoided or limited, but I am not optimistic about it.

In any case, the most that could possibly be guaranteed would be that compressing the unmodified full-color output of djpeg, at the original quality setting, would introduce no further loss. Even such simple changes as cropping off a border could cause further roundoff-error degradation. (If you're wondering why, it's because the pixel-block boundaries move. If you cropped off only multiples of 16 pixels, you might be safe, but that's a mighty limited capability!)

The bottom line is that JPEG is a useful format for archival storage and transmission of images, but you don't want to use it as an intermediate format for sequences of image manipulation steps. Use a lossless format (PPM, RLE, TIFF, etc) while working on the image, then JPEG it when you are ready to file it away. Aside from avoiding degradation, you will save a lot of compression/decompression time this way :-).

• What are some rules of thumb for converting GIF images to JPEG?

As stated earlier, you will lose some amount of image information if you convert an existing GIF image to JPEG. If you can obtain the original full-color data the GIF was made from, it's far better to make a JPEG from that. But if you need to save space and have only the GIF to work from, here are some suggestions for getting maximum space savings with minimum loss of quality.

The first rule when converting a GIF library is to look at each JPEG, to make sure you are happy with it, before throwing away the corresponding GIF; that will give you a chance to re-do the conversion with a higher quality setting if necessary. Some GIFs may be better left as GIFs, as explained in section 5; in particular, cartoon-type GIFs with sixteen or fewer colors don’t convert well. You may find that a JPEG file of reasonable quality will be larger than the GIF. (So check the sizes too.)

Experience to date suggests that large, high-visual-quality GIFs are the best candidates for conversion to JPEG. They chew up the most storage so offer the most potential savings, and they convert to JPEG with least degradation. Don’t waste your time converting any GIF much under 100 Kbytes. Also, don’t expect JPEG files converted from GIFs to be as small as those created directly from full-color originals. To maintain image quality you may have to let the converted files be as much as twice as big as straight-through JPEG files would be (i.e., shoot for 1/2 or 1/3rd the size of the GIF file, not 1/4th as suggested in earlier comparisons).

cjpeg’s default Q setting of 75 is appropriate for full-color input, but for GIF inputs, Q settings of 85 to 95 often seem to be necessary to avoid image degradation.

Many people have developed an odd habit of putting a large constant-color border around a GIF image. While useless, this was nearly free in terms of storage cost in GIF files. It is not free in JPEG files, and the sharp border boundary can create visible artifacts (“ghost” edges). Do yourself a favor and crop off any border before JPEGing. (If you are on an X Windows system, XV's manual and automatic cropping functions are a very painless way to do this.)

Sometimes, smoothing a GIF before compression will reduce the JPEG file size and improve the output image quality. The theory is that smoothing reduces the dithering patterns found in most color GIFs; this helps because dithering creates high-spatial-frequency noise which JPEG doesn’t handle well. If you can see regular fine-scale patterns on the GIF image, then smoothing is definitely indicated. At some point good JPEG software will probably include an input-smoothing option, but for now, you’ll have to use external tools.