InfAI®
Institute for Applied Informatics

JPEG 10
STILL IMAGE CODEC

35 Years of JPEG
and
the Evolution of JPEG 10

On the 7th of October in 1991, 35 years ago, the Independent JPEG Group (IJG) released the first version of libjpeg, a reference software library for JPEG applications. That date marks the start of the digital photographic image era.

This year, in January 2026, the Independent JPEG Group released a groundbreaking new version of the software, JPEG 10, which clarifies the bit depth issue in JPEG.
In the article before last titled “33 Years JPEG: Ubiquitous Presence and Still in its Infancy”, we gave an overview of the new feature and its importance.
We recommend to read this article as an introduction for the following explanation.

> **JPEG 10 was released in January 2026 as a groundbreaking new version.**

In this article, we want to tell the story how the feature which now makes JPEG 10 was discovered. This story also gives the opportunity to trace the evolution from JPEG 8 over JPEG 9 to JPEG 10 as a consistent path.

Some time ago some researchers presented a new kind of JPEG codec with a highlighted new feature described as follows:

> “**10+ bits**. Jpegli can be encoded with 10+ bits per component. Traditional JPEG coding solutions offer only 8 bit per component dynamics causing visible banding artifacts in slow gradients. Jpegli's 10+ bits coding happens in the original 8-bit formalism and the resulting images are fully interoperable with 8-bit viewers. 10+ bit dynamics are available as an API extension and application code changes are needed to benefit from it.”

In fact, everyone who ever implemented a JPEG codec knows that 11 bits are needed for the DCT coefficients in the "original 8-bit formalism" as they call it.
DCT is the Discrete Cosine Transform, core component of the JPEG process, which transforms between the common picture element (pixel) map image representation and the internal harmonic function based representation in the JPEG file.
It is interesting to note that only a vague **10+ bits** statement can be made in above declaration, without an ability to verify the claim.
It was previously believed that the three extra bits are just some kind of precision reserve for the DCT calculation caused by mathematical reasons.
It is really difficult to make any definite claim about preserved bits in the standard 8x8 DCT case due to the relatively complex calculations with 64 values involved in the process.

And now come **JPEG 8** and **JPEG 9** into play.
This is what certain people want to avoid like the plague for various reasons.
And that ignorance becomes the obstacle now for them which prevents a clear comprehension of the issue.

JPEG 8 simply makes the DCT block size variable, one can use anything from 1x1 to 16x16 now, not only the default 8x8 as used *before* JPEG 8.
We call this feature also *SmartScale*, because it is directly related to one of the *fundamental properties for image representation*, which make the substance of JPEG (ignored by everyone else!).
The 1x1 case is basically a no-op (no operation), thus allowing to avoid any calculation and thus **enabling lossless coding within the DCT mode of operation**, which is not possible in the 8x8 case as mentioned above.

JPEG 9 just adds a reversible color transform to make the lossless coding more effective.
At that time we didn't know the feature which now makes JPEG 10, so a "workaround" had to be applied to **save one extra bit in the calculation**.

Another thing which worried us in the lossless coding operation was the fact that a quantization factor of 8 had to be used to achieve the mentioned *no-op* effect in combination with the 1x1 DCT (which is actually a division by 8, or right shift by 3, in the inverse DCT case on the decoder side).
In the normal 8x8 DCT case one would use a factor of 1 for maximum quality.
Using a quantization factor of 8 means that one actually *suppresses* 3 extra bits!
While covered by the calculation effort for the 64 values in the 8x8 DCT case, this fact becomes blatantly obvious in the 1x1 case where only one single value is processed.

Well, it works, but it is strange somehow, and so we kept pondering over the issue.
If 3 bits are somehow *suppressed* in the *lossless* operation, isn't there a way to make use of the extra bits for any reasonable purpose? In particular considering that we needed an extra bit for the reversible color transform? Can we get this extra bit somehow and avoid the workaround?

So this pondering led first to the prospect of the GCbCr color transform for lossless application mentioned in the description in the JPEG 10 source code, but not yet implemented (planned for later...).

Then gradually the full feature has been revealed.

We believe that the feature is already applied in some places, with a vague comprehension similar to the example given above due to the 8x8 cover effect.
The feature is probably applied in digital camera firmwares, with an implementation similar to that in JPEG 10. That explains why JPEG is still commonly used also in high-end digital cameras.

So what is the lesson to be learned from the story?
Study JPEG 8 and JPEG 9, do **not** ignore them!
This is the key to understand JPEG 10, even if only using it in the default configuration.

Notice that JPEG is currently not particularly optimized for the lossless coding case (can be done later if required in the given framework introduced with JPEG 8), it uses just the given coding functions for photos. The projected GCbCr lossless mode will improve compression moderately.

Author:

Guido Vollbeding, Institute for Applied Informatics (InfAI, infai.org),
Independent JPEG Group (ijg.org, jpegclub.org/reference/)

Leipzig, November 2026