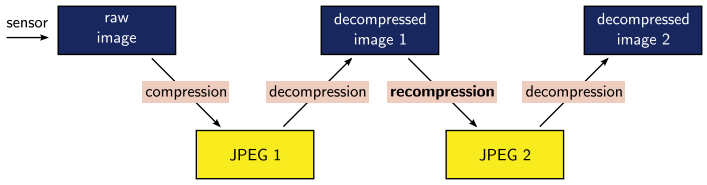
Exact JPEG recompression and forensics using interval arithmetic

Andrew B. Lewis, Markus G. Kuhn

UNIVERSITY OF CAMBRIDGE
Computer Laboratory
Security Group

The JPEG compression algorithm for photographic images is commonly used in digital cameras and web browsers, where it improves storage capacity or transmission speed for photographs by a factor of 10–30.

The standard JPEG compression algorithm was only designed to process images as they were acquired by camera sensors. In practice, however, photographic images are often compressed and decompressed multiple times. Here, image quality can be preserved better with special recompression algorithms designed for input that has been compressed and decompressed before.

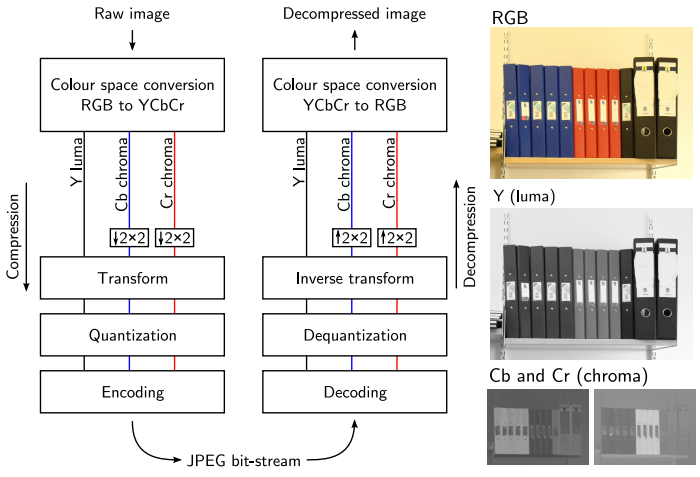


Our new JPEG recompressor represents the exact arithmetic operations of the preceding decompression as an overdetermined system of equations. It then uses interval arithmetic and iterative refinement to invert decompression steps in a way that takes into consideration all possible rounding effects. This results in a set of possible compression results, which will include the actual result of the last compression (and possibly others that if decompressed will result in the exact same raw image).

We have also extended our recompression algorithm into a forensic tool that identifies those regions of a raw image that cannot possibly have been the result of a previous JPEG decompression step.

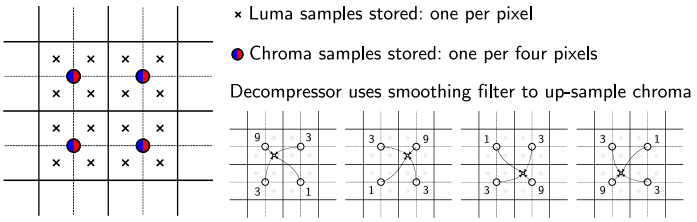
JPEG compression of a colour photograph consists of four lossy transform steps: (a) RGB to $\mathrm{YC_bC_r}$ colour space conversion, (b) $\mathrm{C_bC_r}$ subsampling, (c) discrete cosine transform of 8×8 pixel blocks and (d) quantization.

Equivalently, a decompressor finishes with (a) inverse discrete cosine transform, (b) $\mathrm{C_bC_r}$ interpolation and (c) $\mathrm{YC_bC_r}$ to RGB colour space conversion.

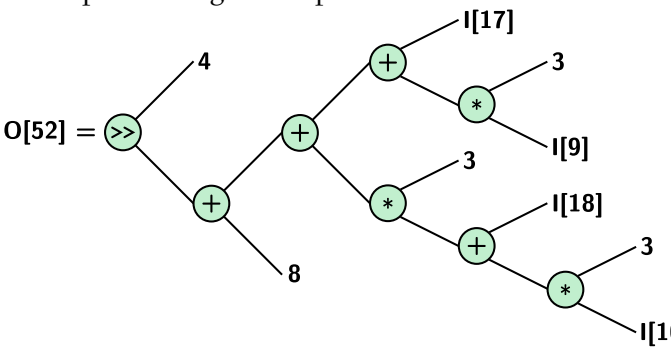


Our exact recompressor inverts the last two of these steps, as implemented in the widely used Independent JPEG Group (IJG) library.

We first build a complete inverted index of the 24-bit RGB to 24-bit $\mathrm{YC_bC_r}$ colour space conversion function in 2^{24} steps. This results in a small (often singleton) set of possible $\mathrm{YC_bC_r}$ values for each RGB pixel.



We then modified the IJG decompressor to output the computational steps of the $\mathrm{C_b}$ or $\mathrm{C_r}$ interpolation as mathematical expression trees. For example, a single pixel $O[52]$ in the $\mathrm{C_r}$ plane is interpolated from four other pixels $I[10]$, $I[18]$, $I[9]$ and $I[17]$ in the subsampled colour plane using this expression:



We now solve the resulting equation in turn for each of the four variables $I[\cdot]$ in the above expression. This results in an interval of possible values for each because:

- we do not know the remainders in the result of integer divisions (i.e., rounding losses);
- the result $O[52]$ is only known to lie within the set of values from the inverted-index lookup;
- we do not (yet) know the values of the other three inputs $I[\cdot]$.

Each input variable $I[\cdot]$ to the above interpolation term appears in four such equations, resulting in a system of overdetermined equations. We initialize all $I[\cdot]$ to the maximum interval $[0, 255]$, and then solve repeatedly for each occurrence of each $I[\cdot]$ in each equation until no further reduction in the size of any interval $I[\cdot]$ can be achieved. The process converges faster if we pass over the image in each possible direction in turn. Applying recompression results in around 4 dB improvement in chroma PSNR for a typical image, and we can localise tampering in test images. In future work, we plan to expand the technique to include the DCT and possibly other decompressor implementations.

2008-06-30