IMAGE COMPRESSION

For articles on related subjects *see* COMPUTER GRAPHICS: STANDARDS; DATA COMPRESSION; IMAGE PROCESSING; MEDICAL IMAGING; and TOMOGRAPHY, COMPUTERIZED.

Digital *image compression* methods reduce the space necessary to encode, store or transmit digital images by changing the way those images are represented. To see why image compression is desirable, consider an 8 × 10 inch color picture, digitized using a resolution of 600 pixels per inch and a byte for each of three color planes. Then the number of bytes required to represent each image is:

$$8 \times 10 \times 600 \times 600 \times 3 \text{ bytes} \approx 85\,\text{MB}.$$

On typical current personal computing systems with only a few gigabytes of hard disk (*q.v.*) space, 85 MB is too much for storage of all but a small number of pictures. Even in the future with much cheaper and larger stores, the ability to compress images by a factor of 25 or more (typical of current standards such as JPEG) will be highly desirable. Similar issues apply to communications.

There are numerous methods for compressing data and each has advantages and disadvantages depending on what kind of image one wishes to compress, how much loss of information one is willing to tolerate, and what form that loss takes. Methods can also be combined. In fact, most standard compression systems combine more than one technique.

Digital Image Representation

The simplest images are bi-level images. This format is often used to encode sharp documents and faxes, where a pixel can assume only one of two values, black or white. Pixel encoding is usually one bit ("0" or "1").

When several gray midtones are present, as in black and white photos, two values are no longer sufficient to encode all possible gray intensities. Thus, each pixel must be assigned a numerical value, proportional to the brightness of that point. Typical choices for these values fall in the ranges 0–15 or 0–255 (requiring respectively 4 and 8 bits for each pixel). This kind of image is referred to as a *gray-level* picture.

Color images take advantage of the fact that each color can be expressed as a combination of three primary colors (red, green and blue or yellow, magenta and cyan, for example). Therefore a color picture can be considered as the superimposition of three "simpler" pictures (called *color planes*), with each of them encoding the brightness of a primary color. In other words, each color plane of an image can be treated much like a gray-level picture with a range of values based on the *luminosity* of that particular color. This type of representation, called RGB or YMC according to the color planes, is "hardware-oriented" (monitors, printers and photographic devices use these color schemes), and it is used when dealing with synthetic (computer-generated) images.

In many scientific applications, images may have more than three planes of information (e.g. multispectral images) and may be higher dimensional. In natural (or non-computer-generated) images, however, the brightness values of corresponding pixels in different color planes are highly correlated (what is bright in the red plane, for example, will often be bright in the green and blue planes) and this correlation can easily be exploited using an alternative system of color representation. Instead of dividing the image into three color planes, the overall brightness of each pixel can be encoded in a luminosity plane (Y). This makes two color planes sufficient to encode the chromatic variations (these two planes are called Cb and Cr). This "YCbCr" color format is used, with some variations, in the broadcast of TV transmissions (the luminance component alone gives a representation that is backward-compatible with black and white (B/W) pictures).

Lossless vs. Lossy Methods

A basic distinction among image compression algorithms can be made in terms of reconstruction fidelity. When it is possible to recover exactly the original data from the compressed data, the algorithm is called *lossless*. Otherwise, when some distortion is introduced in the coding process and part of the original information is irremediably lost, the algorithm is called *lossy*. Lossy compressors, since they can discard part of the information, achieve a more compact representation than lossless systems.

A compromise between these two categories is represented by "near to lossless" or "transparent" algorithms where a small coding error is allowed only when it is semantically irrelevant. These methods are useful, for example, in compressing medical images where two contrasting needs arise: high compression rates (required by the large amount of data) and the preservation of the diagnostic information.

In what follows we focus on lossy methods because lossless methods are covered in the article on DATA COMPRESSION.

Lossy Image Compression

TRANSFORM CODING

Transform coding uses a transformation that exploits peculiar characteristics of the signal, increasing the performance of a scheme such as Huffman or arithmetic coding, for example (*see* DATA COMPRESSION).

For natural sources (audio, images, and video), the information that is relevant for a human user is best described in the frequency domain. When a pictorial scene is decomposed into its frequency components, the low frequencies (gradual luminosity changes) correspond to the scene illumination, and the high frequencies (rapid changes) characterize objects' contours.

Representing the input in the frequency domain results in better control of the error introduced in lossy compression; information that is not important for a human viewer, can be easily discarded or distorted, so that the image will be smaller or easier to compress.

Several time domain to frequency domain transformations have been proposed for digital image signals; one of the most used, the Discrete Cosine Transform (DCT), has the advantage of a good decorrelation of the signal (which reduces redundancy), requiring only an acceptable computational effort.

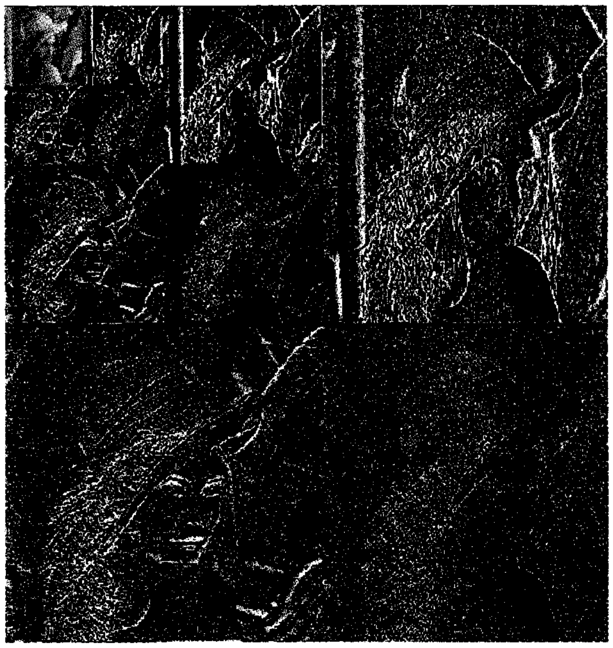


Figure 1. Wavelet decomposition.

Transformations are not compression methods in the literal sense (sometimes they even increase the size of the input), but when used properly they provide a powerful enhancement of entropy-based compression methods. For further reading, see Rao and Yip (1990).

WAVELET COMPRESSION

Another compression scheme that adopts transform coding is based on *wavelet* functions (Fig. 1). The basic idea of this coding scheme is to process data at different scales of resolution. If we look at a picture from a distance, we notice the macro-structure (i.e. the *subject* of the painting). If we move close to the painting, we notice the micro-structures (e.g. the painter's brush strokes). Using wavelet functions allows us to see both the subject and the micro-structure. This is achieved using two versions of the picture at a different scaling of the same prototype function called the *mother wavelet* or *wavelet basis*. A contracted version of the basis function is used for the analysis in the time domain, while a stretched one is used for the frequency domain.

Localization (in both time and frequency) means that it is always possible to find a particular scale at which a specific detail of the signal may be discerned. Wavelet analysis enables the amplitude of the input signal to be drastically reduced. This is particularly appealing for image compression, since it means that half of the data is almost zero and thus easily compressible. For further reading, see Vetterli and Kovacevic (1995).

Vector Quantization

A *dictionary* (or *codebook*) is a collection of a small number of statistically relevant patterns (codewords). Every image is encoded by dividing it into blocks and assigning to each block the index of the closest codeword in the dictionary. Matching can be exact or approximate, thereby achieving, respectively, lossless or lossy compression. The codebook is sometimes allowed to change in response to the input's peculiarities.

The best known dictionary method is *vector quantization*; in its simplest form, it is a lossy compression scheme that uses a static dictionary. A set of images (the *training set*), statistically representative of the source's behavior, is carefully selected; each image is divided into blocks and a small set of dictionary codewords is determined. The codewords are selected to minimize the coding error on the training set. Because of its mathematical tractability, the mean square error is usually used as the error metric and is used to guide both the design and the encoding process (*see* Performance Evaluation below).

The compression rate depends both on the size of each block and on the size of the dictionary. A dictionary with N codewords of $n \times n$ pixels each compresses an image digitized with b bits per pixel with a ratio:

$$R = \frac{n \times n \log_2 b}{\log_2 N}.$$

Once the dictionary is determined, encoding is performed by assigning to each block of the image the index of the closest codeword in the dictionary (i.e. the one with the minimum mean error). The decoder, which also knows the dictionary, simply expands the indices of the codewords into their appropriate blocks.

In a vector quantizer, encoding and decoding are highly asymmetric processes. Searching for the closest block in the dictionary (encoding) is computationally much more expensive than retrieving the codeword associated with a given index (decoding). Even more expensive is the dictionary design, but fortunately this can be done offline.

Vector quantization may be proved to be asymptotically optimal when the size of the block increases; unfortunately the size of the codebook also grows exponentially with the block size.

Many variations of vector quantization have been proposed to speed up the encoding and to simplify the codebook design. Most of these are based on tree (*q.v.*) structures. For further reading see the book of Gersho and Gray (1992).

Fractal Compression

Algorithms based on fractals (*q.v.*) have very good performance and high compression ratios (32 to 1 is not unusual), but their use can be limited by the extensive computation required. The basic idea can be described as a "self vector quantization," where an image block is encoded by applying a simple transformation to one of the blocks previously encoded. Transformations frequently used are combinations of scaling, reflections, and rotations of another block.

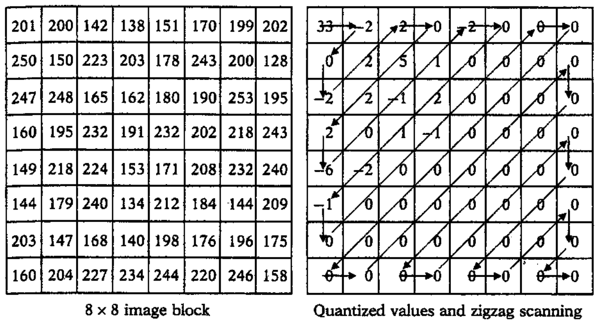
Unlike vector quantization, fractal compressors do not maintain an explicit dictionary, which is why the process can be long and computationally intensive. The basic idea is that each encoded block must be transformed in all possible ways and compared with the current one to determine the best match. Because the blocks are allowed to have different sizes, there is very little "blocking" noise and the perceived quality of the compressed image is usually very good. *See* the books of Barnsley and Hurd (1993) or Fisher (1995).

Data Compression Standards

JPEG

The Joint Photographic Experts Group (JPEG) developed a standard for color image compression that was issued in 1990. The standard was mainly targeted for compressing natural black and white and color images. JPEG, like almost every other transform coding image compression algorithm, defines two steps. The first is lossy and involves transformation and quantization; it is used to remove information that is perceptively irrelevant for a human user. The second step is a lossless encoding that eliminates statistical redundancies still present in the compressed representation.

JPEG assumes a color image divided into color planes and compresses each of them independently. It uses the YCbCr representation discussed earlier, taking advantage of the fact that the human visual system is more sensitive to luminosity than to color changes. Thus it is possible to achieve some compression (even before applying JPEG) just by reducing the resolution of the two chrominance (Cb and Cr) components. Each color plane is further divided into blocks of 8×8 pixels. This size block was determined to be the best compromise between computational effort and the compression achieved. For each block, a decomposition in the frequency domain is computed using the DCT. Fig. 2 shows an 8×8 image block and the result of applying the DCT to it. The result is an 8×8 matrix with some small numbers and several zeros. Using a zigzag pattern, this matrix is scanned from the low to the high frequencies and then converted into a vector. Run-length encoding is applied to compress the sequences of consecutive zeros. The result is then



201	200	142	138	151	170	199	202
250	150	223	203	178	243	200	128
247	248	165	162	180	190	253	195
160	195	232	191	232	202	218	243
149	218	224	153	171	208	232	240
144	179	240	134	212	184	144	209
203	147	168	140	198	176	196	175
160	204	227	234	244	220	246	158

8 × 8 image block

33	-2	2	0	2	0	0	0
0	2	5	1	0	0	0	0
-2	2	-1	2	0	0	0	0
2	0	1	-1	0	0	0	0
-6	-2	0	0	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

Quantized values and zigzag scanning

Figure 2. JPEG: an image block and a block showing the quantized values after applying the DCT as well as the zigzag pattern used in scanning it.

further compressed using a Huffman or an arithmetic coder (*see* DATA COMPRESSION).

JPEG-LS

In 1994 ISO/JPEG issued a call for a new standard for lossless and near-lossless compression of continuous tone images (2 to 16 bits per pixel—bpp). This standard, called JPEG-LS, is a predictive coder which uses an error-feedback technique. The encoder classifies the prediction context and stores the mean prediction residual that occurs in each class. After the prediction step, the encoder locates the class to which the current image context belongs and adds the mean error to the prediction. The prediction residual is then computed and coded.

In addition to providing a better prediction, the advantage of the error feedback technique is that errors in different contexts have different probability distributions which can be used to tune the coder to the specific context. *See* Weinberger *et al.* (1996).

JBIG

In 1991 the Joint Bi-level Image Experts Group (JBIG) defined an innovative lossless compression algorithm for black and white images. It is a predictive coder that uses a pool of already coded neighbor pixels to guess the value of the current pixel. The algorithm simply concatenates the value of the template pixels to identify the context in which the current pixel is going to be predicted. The index of the context is used to choose which probability distribution should be used by an arithmetic coder.

The "A" pixel in Fig. 3 is an *adaptive* pixel. Its position is allowed to change as the image is processed. The use of an adaptive pixel improves compression by identifying repeated occurrences of the same block of information.

JBIG may also be successfully used in coding images with more than one bit per pixel for gray-scale or color images. The image is decomposed into *bit-planes*

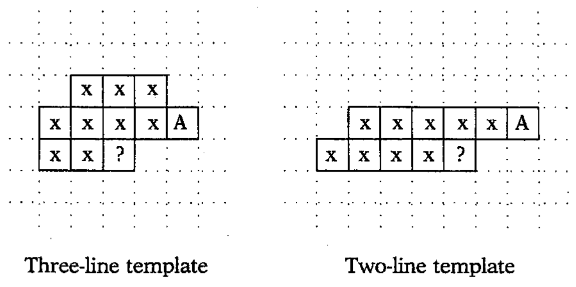


Figure 3. Prediction templates for JBIG.

(i.e. if the image is 4 bpp, each pixel is represented by the binary string $b_3 b_2 b_1 b_0$; bit-plane i then stores bit b_i of each pixel) and each plane is coded separately. In principle, JBIG could work with 255 bpp images, but in practice the algorithm has good performance only for images with at most 8 bpp. *See* Arps and Truong (1994).

Performance Evaluation

The performance of an image compression algorithm is mainly determined by two characteristics: the *compression ratio* and the *magnitude of the error* introduced by the encoding. In a lossless compressor, the size of the image is minimized while retaining the quality of the original. But a lossy algorithm must compromise between these two qualities.

A fundamental problem in lossy compression is controlling the error introduced by encoding process. Among several quality metrics that are commonly used the mean square error (MSE) criterion is much the most important. The MSE between a given image $i(x, y)$ and its encoded version $\hat{i}(x, y)$ is the square root of the sum of the squares of the differences between the corresponding values of the samples in the two signals:

$$\text{MSE} = \sqrt{\sum_{x,y} (i(x, y) - \hat{i}(x, y))^2}.$$

The MSE gives a good measure of the random error introduced in the compression; this is enough for many applications, but, when encoded images are mainly used by humans, the use of distortion measures based on the MSE may give misleading results. The poor correlation of the MSE with the perceived distortion occurs because the human visual system is more sensitive to structured than to random coding errors. *See* the book of Gonzalez and Wintz (1987).

Coding Artifacts

When a very high compression rate has to be achieved or when a complex image has to be encoded, lossy compression methods sometimes introduce visible artifacts that can make the perceived quality very poor. Commonly observed artifacts are:

- *Blocking*, which happens when a gradual change in the intensity or color of a region is coarsely quantized. This results in periodic discontinuities in the image, which appears segmented into its constituent blocks.
- *Blurring*, which appears in different forms such as at edges, due to the loss of high-frequency components, or as blurring of the texture and color due to the loss of resolution.
- *Ringing effect*, which is observable as periodic pseudo-edges around original shape edges for the compressed images. The ringing effect results from improper truncation of high-frequency components, also known as the Gibbs effect.
- *Texture deviation*, which appears as granular noise or as the "dirty window" effect, and it is caused by loss of fidelity in mid-frequency components.

See Woods (1991).

Bibliography

1987. Gonzalez, R., and Wintz, P. *Digital Image Processing.* Reading, MA: Addison-Wesley.
1990. Rao, D., and Yip, P. *Discrete Cosine Transform.* San Diego, CA: Academic Press.
1991. Woods, J. *Subband Image Coding.* Norwell, MA: Kluwer Academic Publishers.
1992. Gersho, A., and Gray, R. M. *Vector Quantization and Signal Compression.* Norwell, MA: Kluwer Academic Publishers.
1993. Barnsley, M., and Hurd, L. *Fractal Image Compression.* Natick, MA: AK Peters.
1994. Arps, R. B., and Truong, T. K. "Comparison of International Standards for Lossless Still Image Compression," Special Issue on Data Compression (ed. J. Storer). *Proceedings of the IEEE,* **82**, *6*, 889–899.
1994. Kondoz, A. *Digital Speech.* New York: John Wiley.
1995. Bhaskaran, V., and Konstantinides, K. *Image and Video Compression Standards.* Boston: Kluwer Academic Press.
1995. Fisher, Y. (ed.) *Fractal Image Compression: Theory and Application.* New York: Springer-Verlag.
1995. Vetterli, M., and Kovacevic, J. *Wavelets and Subband Coding.* Upper Saddle River, NJ: Prentice Hall.
1996. Sayood, K. *Introduction to Data Compression.* San Francisco: Morgan Kaufmann.
1996. Weinberger, M. J., Seroussi, G., and Sapiro, G. "LOCO-I: A Low Complexity, Context-based, Lossless Image Compression Algorithm," *Proceedings IEEE Data Compression Conference,* Snowbird, UT, 140–149.
1997. Haskell, B. G., Puri, A., and Netravali, A. N. *Digital Video: An Introduction to MPEG-2.* New York: Chapman & Hall.
1997. Mitchell, J., Pennebaker, W., Fogg, C., and LeGall, D. *MPEG Video Compression Standard.* New York: Chapman & Hall.
1997. Salomon, D. *Data Compression: The Complete Reference.* New York: Springer-Verlag.

Website

Computer compression newsgroup FAQ. `http://www.faqs.org/faqs/compression-faq/part2/`.

G. Motta, F. Rizzo, and J. A. Storer