2D Pattern Repetition Based Lossless Image Compression

Dénes Paczolay

Recently several lossless compression methods were proposed for low-gradient (continuous-tone) images. The majority of everyday photos, and also medical images and satellite images tend to fall into this category. These algorithms were developed for grey-scale images but, analogously, they can be applied to the colour components of coloured images as well. The efficient lossless image compression algorithms developed for low-gradient images are based on the coherence of pixel intensity. That is, based on the intensity of the neighboring pixels they are able to give a good estimate of the intensity of the actual pixel. [4] The more peaked the Gaussian distribution error, the more efficient the estimation will be. It is usually easier to compress the resulting error-image, which is the difference between the original image and the estimated pixel intensities for it. The estimation uses a few parameters at most, so we can get a good compression ratio. An earlier trend was to apply lossy image compression for the purpose of estimation, but the precision of the estimate obtained this way greatly depended on the size of the lossy compressed image, and in many cases this method resulted in worse, rather than better compression ratios [3].

In practice not only low-gradient images need to be compressed, but also high contrast or colour palette images. In such cases, it frequently occurs that the pixels cannot be efficiently estimated from their neighbours, especially when the pixels represent colour indices. But even in these cases the image may contain repetitions which can be exploited for efficient compression. The algorithm which was developed here for such images is based on traditional and general compression techniques. For example, the GIF format uses LZW [1] and the PNG format uses LZ77 [2], these algorithms actually forming part of a well-known zip packer. To improve the compression ratio on low-gradient images the PNG compressed image format uses some simple filters (estimations) – which may be different for each row. A common feature of these widely used compression methods is that the compression and search is done one line at a time. Interestingly, a common "deficiency" of all these methods is that if we rotate the image by 90 degrees, we can get a noticeably different compressed image size.

The novel lossless image compression algorithm described in this paper was developed for images that contain repetitions of identical regions. Such images are produced when using remote desktop access (rdesktop), virtual network computing (VNC), document faxing, or document-to-image conversion. The algorithm we propose here compresses not just the homogenous areas, but also repeated background patterns, so it works very efficiently for patterned background documents and the screenshots of a web-page. If the image contains many texts (take, for example, a screen shot of this abstract), the method finds the blocks of the characters and words and then compresses them.

When searching for the repetitions of 2D patterns, we have to solve the problem of handling overlapping regions. For this we can store information about which pixels have already been processed (compressed), so at these points the equivalence of pixels does not have to be examined. Because the regions tend to overlap the compression algorithm processes the pixels in a non-linear order. A further optimization step we might take is to determine the optimal number of bits required for storing the region sizes and the search distance; we can separate large and small regions if necessary. The pixels that are not compressed (for example, the first pixel) can be stored using Huffman Coding, Arithmetic Coding or some other way. Huffman Coding is simple, but requires storing the Huffman table for decoding or the usage of a dynamic Huffman Code. In experiments we found that in some cases it is better to use a "pixel cache" than some other method. The "pixel cache" contains the recently used 2^m pixel values (or colours). If the next uncompressed pixel is in the "cache" we store its index using $m+1$ bits, otherwise we store the pixel or colour using $n+1$ bits and add this pixel to the "cache" (where the original image stored the pixels in n bits and $m+1<n$). If necessary, we can separate the

short and long series of stored pixels and hence optimize the number of bits needed to show each storage block (short/long stored pixels, small/large repeated region).

Our algorithm usually proves more efficient than PNG or GIF on the sort of images described above. Furthermore, similar to PNG, our algorithm is suitable for palette and RGB images as well.

References

[1] T.A. Welch. A Technique for High Performance Data Compression, *IEEE Computer*, 17, 6, 8-19, 1984.

[2] J. Ziv and A. Lempel. A Universal Algorithm for Sequential Data Compression, *IEEE Transactions on Information Theory*, 23, 3, 337-343, 1977, citeseer.ist.psu.edu/ziv77universal.html.

[3] X. Wu. An Algorithmic Study on Lossless Image Compression, *Data Compression Conference*, 150-159, 1996, citeseer.ist.psu.edu/wu96algorithmic.html.

[4] N.V. Boulgouris, D.Tzovaras and M.G. Strintzis. Lossless Image Compression Based on Optimal Prediction, Adaptive Lifting and Conditional Arithmetic Coding, *IEEE Trans Transactions on Image Processing*, 10, 1, 1-14, 2001.