On the Selection of Image Compression Algorithms - Review

Buzdugan Alexandru Marcel

Introduction

After lecturing the paper published by Chaur-Chin Chen and the Department of Computer Science from Tsing Hua University and researching the topic it is clear that the field of data compression and especially image compression is a complex and difficult one.
They started with the premise that there is no perfect compression algorithm and that all of them have flaws and some of them shine brighter than others under different conditions. They reviewed a number of 4 compression methods using 4 256x256 images that are quite common for this task (Jet, Lenna, Mandrill and Peppers) and one 400x400 fingerprint image. Please keep in mind that the paper is quite old and some of these methods were very good on their time but nowadays there are other ones that are commonly used. This field like most others is based on building on the work of their predecessors and that is what has been done.
Those methods display basic principles for image compression that are used in almost all other ones. As a target they had in mind a 0.5 bpp compression rate but they tried to reach 0.25 bpp and saw how the methods reacted then.
They concluded that one of them performed the best and is most likely the first choice for most software developers but the performance of others is also to be noted.

About compression

Data compression is the process of reducing the size of data. This is helpful when we want to reduce the storage and decrease the transmission time when sending resources to other systems.
Data compression is subject to a space–time complexity trade-off which means that we have to think if it is worth the trade of time as computation cycles for storage as RAM, HDD, SSD.
We can apply this process to different types of data like images, audio and video.

Regarding the classification of such methods there are 2 of them, lossy and lossless compression.
Each one of them reduces the size of data, the difference is how they do it. Lossy compression is a method of dropping unimportant data that the human eye does not care very much about while the lossless one removes the statistical redundancy in the picture.

The second one has a reversible process which means that we will not lose data and can reconstruct the image back whenever we want. Bothe types contain multiple methods and are widely used, lossy compression in digital cameras and DVD's and lossless in zip, music, and pretty much everywhere where is important that the original data is the same as the decoden one without any data loss.

Lossless compression methods :
- PNG – Portable Network Graphics
- TIFF – Tagged Image File Format
- WebP – (high-density lossless or lossy compression of RGB and RGBA images)
- BPG – Better Portable Graphics (lossless/lossy compression based on HEVC)
- FLIF – Free Lossless Image Format
- JPEG-LS – (lossless/near-lossless compression standard)
- TGA – Truevision TGA
- PCX – PiCture eXchange
- JPEG 2000 – (includes lossless compression method, as proven by Sunil Kumar, Prof San Diego State University[citation needed])
- JPEG XR – formerly WMPhoto and HD Photo, includes a lossless compression method
- ILBM – (lossless RLE compression of Amiga IFF images)
- JBIG2 – (lossless or lossy compression of B&W images)
- PGF – Progressive Graphics File (lossless or lossy compression)

Lossy compression methods :
- Better Portable Graphics, also known as BPG (lossless or lossy compression)
- Cartesian Perceptual Compression, also known as CPC
- DjVu
- Fractal compression
- ICER, used by the Mars Rovers, related to JPEG 2000 in its use of wavelets
- JBIG2 (lossless or lossy compression)
- JPEG
- JPEG 2000, JPEG's successor format that uses wavelets (lossless or lossy compression)
- JPEG XR, another successor of JPEG with support for high dynamic range, wide gamut pixel formats (lossless or lossy compression)
- Vector quantization (VQ compression)
- PGF, Progressive Graphics File (lossless or lossy compression)
- S3TC texture compression for 3D computer graphics hardware
- Wavelet compression

We can see that all of the methods that the authors tested are lossy since that was the popular choice back then. Nowadays the industry is inclined more toward lossless ones. That should not be taken as the latter being better than the first since both types have their application and do better in different scenarios.

JPEG compression

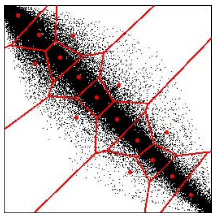
This is one of the most popular method in image compression and is the first choice for our authors. The process is comprised of 5 steps, each playing its role and applying a set of changes on the image. Before starting the process, we have to use the proper color space. We'll have to convert from the usual one which is RGB(Red, Green and Blue) to YCbCr(Y-Intensity of light, Cb - color blue, Cr - color red).
After doing so the next phase begins where we have to split the image in 8*8 blocks (64 pixels) and apply a function on each block for each pixel that subtract from each pixel value 127. This is a preprocessing step, preparing the image for the application of the DCT(Discrete cosine function) in the next step.
A discrete cosine transform (DCT) expresses a finite sequence of data points in terms of a sum of cosine functions oscillating at different frequencies. DCTs are important to numerous applications in science and engineering, from lossy compression of audio (e.g. MP3) and images (e.g. JPEG) (where small high-frequency components can be discarded), to spectral methods for the numerical solution of partial differential equations.
In this step by doing so we are getting rid of the high frequency signal in the picture. The human eye is not very sensible to such frequencies and they don't matter that much in an image.
The 4th step is comprised from the application of 2 quantization table that we will be multiplying with each block of pixels. This will prepare the image for the final step, which is the encoding step. Character encoding is used to represent a repertoire of characters by some kind of encoding system. Usually the Huffman coding is used here which will produce the final result a table that will contain for each pixel values it's frequency. The same process will be repeated for decoding thus meaning that both operations will take the same amount of time.

Wavelet compression

The second choice of our authors is yet another popular method greatly appreciated by the academic community. This one has at its core the mathematical properties of the wavelet. A wavelet is a wave-like oscillation with an amplitude that begins at zero, increases, and then decreases back to zero. It can typically be visualized as a "brief oscillation" like one recorded by a seismograph or heart monitor. It makes use of the wavelets theory which states that the wavelets forming a continuous wavelet transform (CWT) are subject to the uncertainty principle of Fourier analysis respective sampling theory: Given a signal with some event in it, one cannot assign simultaneously an exact time and frequency response scale to that event. The product of the uncertainties of time and frequency response scale has a lower bound. This means that by having only a small portion of the wavelet we can deduce the rest. This process has 3 steps that are initiated by an application of the DCT on an image. This will produce 2 other images (the horizontal approximation and the horizontal detail). Then the same function will be applied for each image thus resulting in another 4 images. By removing the values close to 0 from those images we obtain a smaller image than before. The decoding process is the same as for encoding but backwards.

VQ compression

The third choice is the vector quantization method. This one is a classical quantization technique from signal processing that allows the modeling of probability density functions by the distribution of prototype vectors. It works by dividing a large set of points (vectors) into groups having approximately the same number of points closest to them. Each group is represented by its centroid point, as in k-means and some other clustering algorithms. The first step of this process is to partition the image in blocks of size m*m where m is 4 by default. Then we have to represent those blocks as vectors of m*m tuples called training vectors. The goal here is to obtain few representative vectors called code vectors that will be used for encoding and decoding. The encoding procedure is to look for a closest code vector in the codebook for each non overlapped block of an image to be encoded. The most important work is to design those code blocks. This method has many advantages because is contains a simple decoder, no coefficient quantization, blinding fast decompression and very good quality. Unfortunately the creation of the codebook is quite expensive and can take some time.

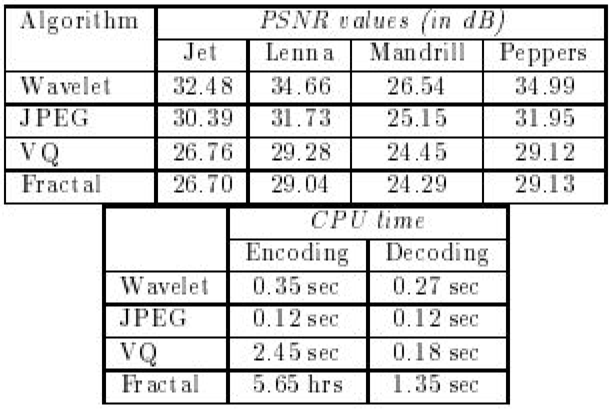

A representation of the codebook vectors using a Voronoi diagram

Fractal compression

The 4th and final one is one method that makes use of fractals, it's called the Fractals compression method. The method is best suited for textures and natural images, relying on the fact that parts of an image often resemble other parts of the same image. Fractal algorithms convert these parts into mathematical data called "fractal codes" which are used to recreate the encoded image. A challenging problem of ongoing research in fractal image representation is how to choose the $f1,...,fN$ such that its fixed point approximates the input image, and how to do this efficiently. To do that we have to take 3 steps by starting with first partitioning the image into *range blocks* of size s*s. After that we have to find blocks of size 2x*2x that are very similar to R_i. Those are marked as D_i.
The final step is to find a functions such that H(Di) = Ri for each i. In the second step, it is important to find a similar block so that the IFS accurately represents the input image, so a sufficient number of candidate blocks for Di need to be considered. On the other hand, a large search considering many blocks is computationally costly. This bottleneck of searching for similar blocks is why PIFS fractal encoding is much slower than for example DCT and wavelet based image representation. This method has a good mathematical encoding frame and a resolution free decoding but is kinda slow. It is really similar with the vector quantization method.

Experimental results

The rating of such an algorithm is measured by calculating the PSNR(Peak signal to noise ratio). This value represents how alike the decompressed image is compared to the original one. The produced results are displayed in dB (bit depth) which is equals to 8 bits.

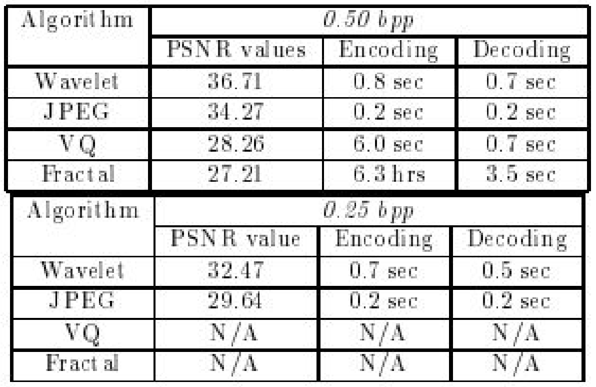


Algorithm	*PSNR values (in dB)*			
	Jet	Lenna	Mandrill	Peppers
Wavelet	32.48	34.66	26.54	34.99
JPEG	30.39	31.73	25.15	31.95
VQ	26.76	29.28	24.45	29.12
Fractal	26.70	29.04	24.29	29.13

	CPU time	
	Encoding	Decoding
Wavelet	0.35 sec	0.27 sec
JPEG	0.12 sec	0.12 sec
VQ	2.45 sec	0.18 sec
Fractal	5.65 hrs	1.35 sec

Table 2: Performance of coding algorithms on various 256×256 images.

After running the methods on the images mentioned in the beginning those are the results they produced.



Algorithm	*0.50 bpp*		
	PSNR values	Encoding	Decoding
Wavelet	36.71	0.8 sec	0.7 sec
JPEG	34.27	0.2 sec	0.2 sec
VQ	28.26	6.0 sec	0.7 sec
Fractal	27.21	6.3 hrs	3.5 sec

Algorithm	*0.25 bpp*		
	PSNR value	Encoding	Decoding
Wavelet	32.47	0.7 sec	0.5 sec
JPEG	29.64	0.2 sec	0.2 sec
VQ	N/A	N/A	N/A
Fractal	N/A	N/A	N/A

Table 3: Performance of coding algorithms on a 400×400 fingerprint image.

This is another interesting find since we can see that they tested the algorithms to perform compression under 0.50 (bits per pixel) and all of them produced results in a fairly reasonable time. We can see clearly here the advantages and the drawbacks of the concepts used for each method. Under 0.25 bpp only 2 of them managed to produce results the other 2 being unable to compress so low.

Conclusion

Looking at the results the authors drew some conclusions regarding each method and of course that as we can also see the Wavelet transform method produced the best results and is the number 1 choice among those 4.
They drew 5 conclusions and those are as follows:

- Wavelet based compression algorithms are strongly recommended.
- DCT based approach might use an adaptive quantization table.
- VQ approach is not appropriate for a low bit rate compression although it is simple.
- Fractal approach should utilize its resolution free decoding property for a low bit rate compression.
- Hybrid compression algorithms based on some of these four approaches may be pursued to achieve a higher compression ratio while preserving a better quality of uptodate decoded.

As a personal conclusion I also think that by looking at the algorithms the wavelet transform is the strongest from a mathematical point of view and it consumes the least resources out of them. Since it makes use of the wavelet properties and it does not require other helping data la tables or vectors that have to be deduced or computed it is also quite fast while also producing the best results. Indeed the Discrete cosine transformation (JPEG) can produce even better results than those displayed here by using an adaptive quantization table and not the standard one. The vector quantization approach produces better results as the images grew because it produces the initial codebook of vectors that then can be used to reconstruct bits of similar blocks of the image. The fractal compression could also produce better results if it had utilized it’s resolution free decoding property for a low bit rate compression.
Those being said i consider this study a good study when it comes to choosing an compression method and certainly everybody interested in developing such methods can make use of this paper.

Sources used for writing this review:

- https://en.wikipedia.org/wiki/Data_compression
- https://en.wikipedia.org/wiki/Space%E2%80%93time_tradeoff
- https://en.wikipedia.org/wiki/Lossless_compression
- https://en.wikipedia.org/wiki/Lossy_compression
- https://en.wikipedia.org/wiki/Lossy_compression
- https://en.wikipedia.org/wiki/Lossless_compression
- https://www.youtube.com/watch?v=dSi9mLaa-WE&t=161s
- https://pdfs.semanticscholar.org/a2e8/3afe88ad1eae88f11585260c4e7b9aff2ec9.pdf
- https://medium.com/@danojadias/jpeg-compression-algorithm-969af03773da
- https://en.wikipedia.org/wiki/YCbCr
- https://en.wikipedia.org/wiki/Variable-length_code
- http://www.ti.com/lit/an/bpra065/bpra065.pdf