AN IMAGE COMPRESSION SCHEME BASED ON LAPLACIAN PYRAMIDS

Catalin Ispas[1*]
Costin-Anton Boiangiu [2]

ABSTRACT

In this paper, we propose an image compression scheme based on the Laplacian image pyramids. First, the image is split into four sub-images repeatedly, in order to find an optimal split position based on their variance. The split point slides between a starting point and ending point and it's stored at every step. After finding the optimal split point, the four sections determined by it are used to build four image pyramids, one for each sub-image. The data of each pyramid is stored in a custom file format and is compressed using BZ2.

KEYWORDS: *image compression, image resampling, summed area tables, Laplacian pyramid, bzip2, Lanczos filter, generalized pyramid*

1. INTRODUCTION

Image compression has become an important subject nowadays, with the increase of content that can be found on websites or on the Internet, such as static images or videos and with the advent of mobile devices and video streaming. The desire to make an eye-catching website also brings the problem of transmitting that content in a timely manner, since users don't like to wait for it to be delivered most of the times or expect the content to be delivered in an imperceptible timeframe.

Laplacian pyramids as means of image compression were introduced by Peter J. Burt and Edward H. Adelson, in the paper "The Laplacian Pyramid as a Compact Image Code" [4]. A Gaussian pyramid (figure 1) is built by repeatedly downsampling the original image, then the Laplacian pyramid is constructed by calculating the difference between the image on level L of the Gaussian pyramid and the upsampled version of the one at level L+1 [1].

Each error image resulting out of this difference is a level in the Laplacian pyramid. Compression is achieved by quantizing the pixel values in the error images. The original image can be recovered by upsampling and summing all the levels of the Laplacian pyramid [4].

[1*] corresponding author, Engineer, "Politehnica" University of Bucharest, 060042 Bucharest, Romania, ispas.catalin@ymail.com
[2] Professor PhD Eng., , "Politehnica" University of Bucharest, 060042 Bucharest, Romania, costin.boiangiu@cs.pub.ro

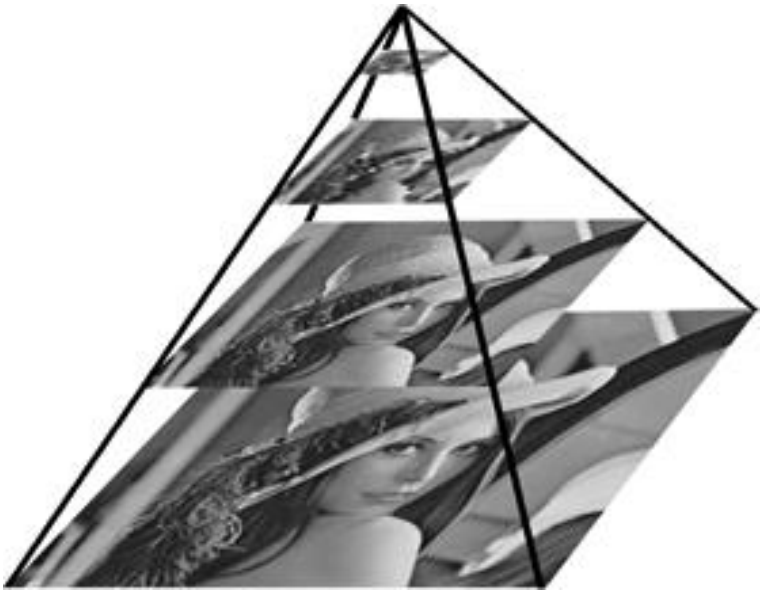

Figure 1. Graphical representation of a Gaussian pyramid

For image compression, a similar scheme was proposed by Costin Anton Boiangiu et al. in the paper "A Generalized Laplacian Pyramid Aimed at Image Compression" [6], where a scanning pattern is used to traverse the input image during the processing phase, in order to group similar pixels, which might help to obtain better compression, when a residual encoding algorithm such as Run-Length Encoding is used. Then a cross point is sought, to create four sections of the input image, which will be utilized to construct four Laplacian pyramids.

The concept of using Laplacian pyramids for image compression is further expanded for video data, by Adrian Enache and Costin Boiangiu, in the paper "A Pyramidal Scheme of Residue Hypercubes for Adaptive Video Streaming" [5], where „hypercubes are built as residues between successive downsampling and upsampling operations over chunks of video data".

This paper proposes an image compression scheme based on splitting the original image into four sub-images, each encoded into its corresponding Laplacian pyramid. Splitting the image serves the purpose of separating the "negative spaces", in order to obtain better compression.

2. THE PROPOSED METHOD

The first step of this proposed method is to search for an optimal point inside the input image, in order to split it into 4 sub-images. The search process starts off in the upper left corner of the image and ends in the lower right corner. The start point p_s and end point p_e are defined as in (1) and (2), where w is the width and h is the height of the image:

$$p_s(x,y) = \left(\frac{w}{4}, \frac{h}{4}\right) \quad (1)$$

$$p_e(x,y) = \left(w - \frac{w}{4}, h - \frac{h}{4}\right) \quad (2)$$

to avoid special cases in which a sub-image is too small to make any additional operations on it or its size is zero.

Once these points are obtained, the input image is traversed from left to right, top to bottom. At each step, four sub-images are formed and the variance value for each is calculated using the formula:

$$v_j = \sum_i (x_i - \mu_j)^2 \quad j = 1..4 \quad (3)$$

$$\mu_j = \frac{\sum_i x_i}{n_j} \quad (4)$$

where μ_j and n_j are the mean value and number of pixels of sub-image "j". The product of these four values is calculated and stored for later use. Due to performance reasons, a summed area table (integral image) was used to compute the four variance values (it's also worth noting that the input image is treated as a 1D array).

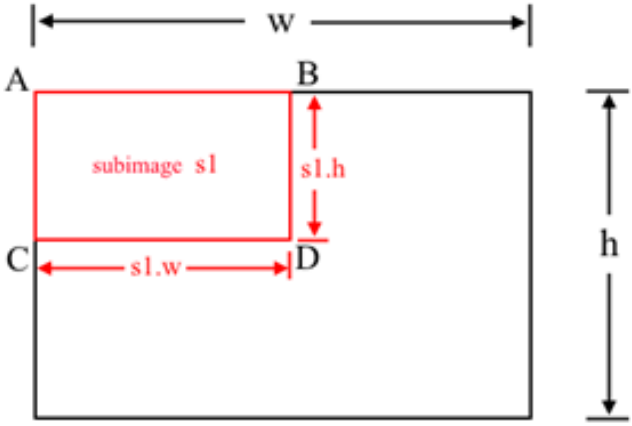

Figure 2. Calculating the sum $\sum_i x_i$ (represented as S) for a sub-image s1, by using the values in the summed area table T, recovered from the indices A, B, C and D (where w and h are the width and height of the input image, s1.w and s1.h are the width and height of the sub-image s1, s1.x and s1.y are the (x, y) coordinates of the top left corner of sub-image s1 and T is the summed area table)

Two tables are pre-computed from the input image: an integral image for simple sums and one for squared sum values. Instead of pixel intensity values, an integral image contains values which are the "the sum of the intensities of all pixels contained in the rectangle defined by the pixel of interest and the lower left corner of the texture image" [2]. For

example, to determine the sum $\sum_i x_i$ from relation (4) for a particular sub-image, only a sum and two differences of values from the summed area table are necessary (see Figure 2).

After the traversal has finished, the lowest variance product from the ones that were stored is utilized to pick the optimal split point (see Figure 3).

Figure 3. Sub-images resulting from splitting the input image at the optimal split point

The next step, once the optimal split point has been found, is to generate four Laplacian pyramids, one for each sub-image. In the case of one pyramid, the sub-image is first downsampled (and then upsampled) and the difference between the original image and the resampled one is computed (figure 4). This procedure is repeated until the size of the sub-image has reached 1 pixel (or the compression scheme has become too inefficient, due to the fixed data overhead added on every stored level) then the process stops.

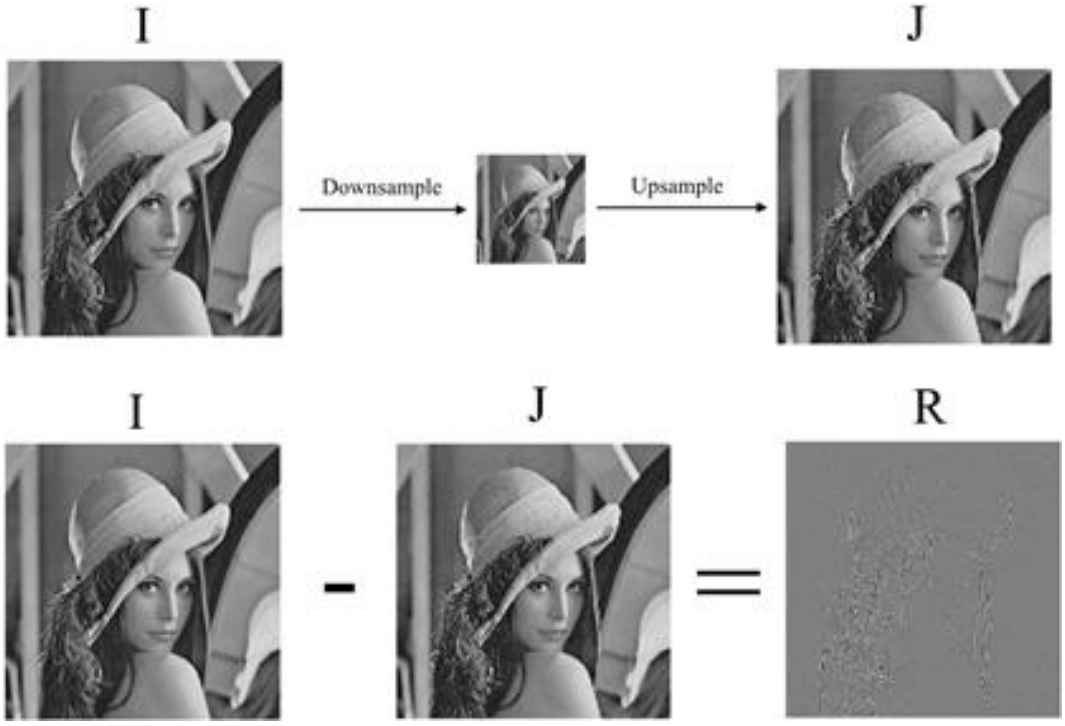

Figure 4. Obtaining a residue image R from the images I and J (produced by downsampling and upsampling I), this represents one level of the Laplacian pyramid

The motivation behind this step is that, although more sample elements that in the input image are produced, most of the sample values "tend to be near zero, and therefore can represented with a small number of bits" [3]. BZ2 [8] was chosen to compress the error images which resulted after this step, because one of the compression techniques it uses is the Burrows-Wheeler transform [9], which further improves compression when it comes to repeating sequences of values, which might exist in this case. No further processing is done on the residues, each row of an individual error image is stored in the order described by a raster scanning pattern.

Custom file formats

Two custom file formats were used: one for the image pyramid and one for storing all the resulting four image pyramids. The PIFF (Pyramid Image File Format) contains information such as: the number of pyramid levels, an array with the widths and heights of the residues found at each level, an array containing the pixel data of all the residues and the array's number of elements. Because the error images resulted during the pyramid construction phase can contain negative values, their pixel data is stored as short integers. The residue dimensions are kept in the following manner: odd index values contain the heights and even index values the widths of the residues (Figure 5).

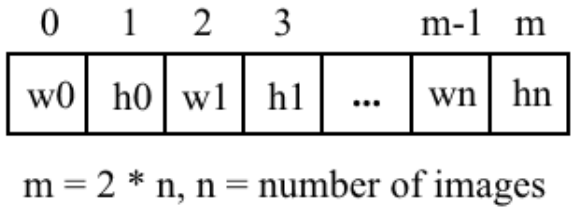

Figure 5. Graphical representation of the array in which the residues dimensions are stored

The MPIFF (Multi-Pyramid Image File Format) contains the original width and height of the input image and four pyramid structures like the one described earlier (Figure 6).

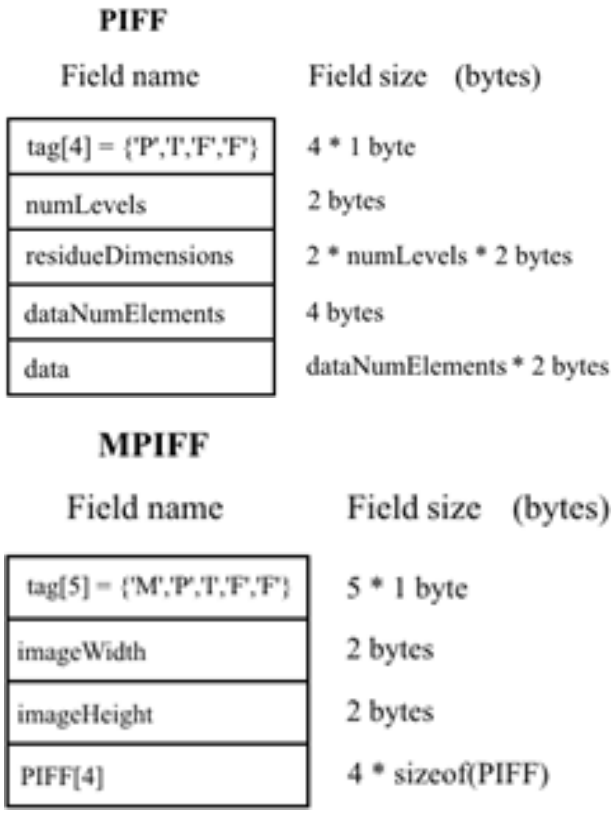

Figure 6. The structure of the Pyramid Image File Format (up) and the Multi-Pyramid Image File Format (down)

4. OBTAINED RESULTS

During the testing phase, four grayscale versions of the following images were used: Lena, Baboon, Peppers and Jellybeans [7], all in uncompressed format (figure 7). For decimating the images, during the image pyramid construction phase, a scaling factor of 4 was used and 3 types of filters were tested: Nearest Neighbour, Cubic and Lanczos. The compression algorithm used is BZ2.

Table 1. Performance of the proposed image compression scheme

Image name	Resolution	Raw size	Filter	Compression (bytes)	Compression factor
Lena	512x512	257 kb	Nearest neighbour	207 955	1.2703132
			Cubic	183 140	1.4424374
			Lanczos	184 334	1.4330942
Baboon	512x512	257 kb	Nearest neighbour	252 785	1.0450303
			Cubic	238 876	1.1058792
			Lanczos	240 100	1.1002415
Peppers	512x512	257 kb	Nearest neighbour	210 266	1.2563514
			Cubic	189 912	1.3910021
			Lanczos	191 396	1.3802169
Jellybeans	256x256	65.7 kb	Nearest neighbour	37 477	1.7975825
			Cubic	33 148	2.0323398
			Lanczos	33 868	1.9891342

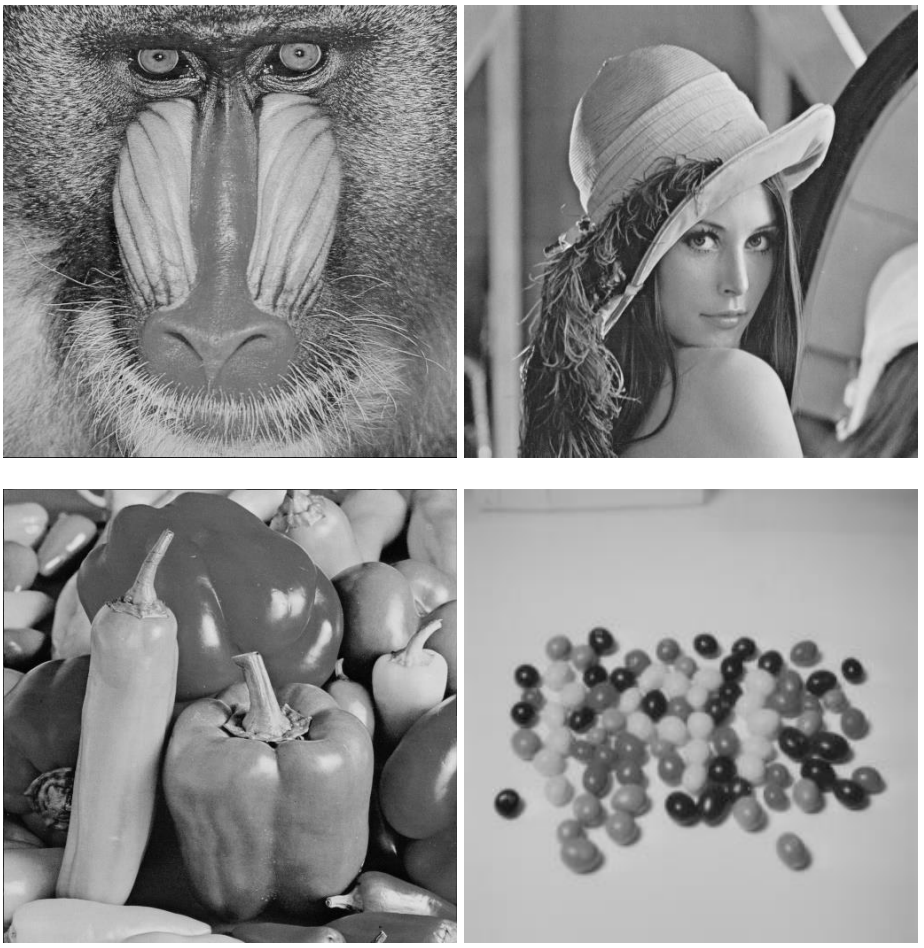

Figure 7. The four images used to test our compression scheme (left to right, top to bottom): Baboon, Lena, Peppers, and Jellybeans (*Source: The USC-SIPI Image Database*)

5. CONCLUSION

In this research paper, an image compression scheme based on Laplacian pyramids was proposed. The best results were obtained on images that contain negative spaces, such as the image "Jellybeans". The algorithm could be further improved by introducing space filling curves, which could help attain better compression, by grouping similar pixels [6]. Also worth trying is finding a better way of encoding negative values, rather than using short integers to store negative values.

REFERENCES

[1] Jeff Perry, Image compression using Laplacian pyramid encoding, C/C++ Users Journal, volume 15, issue 12, pp. 35-47, Dec. 1997.

[2] Franklin C. Crow, Summed area tables for texture mapping, Computer Graphics, volume 18, number 3, pp. 207-212, July 1984.

[3] E. H. Adelson, C. H. Anderson, J. R. Bergen, P. J. Burt , J. M. Ogden, Pyramid methods in image processing, RCA Engineer, Volume 29-6, pp. 34-41 Nov/Dec 1984.

[4] Peter J. Burt, Edward H. Adelson, The Laplacian Pyramid as a Compact Image Code, IEEE Transactions on Communications, volume 31, issue 4, pp. 532-540, April 1983.

[5] Adrian Enache, Costin Boiangiu, A Pyramidal Scheme of Residue Hypercubes for Adaptive Video Streaming, International Journal of Computers and Communications, volume 8, pp. 128-133, 2014.

[6] Costin Anton Boiangiu, Marius Vlad Cotofana, Alexandru Naiman, Cristian Lambru, A Generalized Laplacian Pyramid Aimed at Image Compression, Journal of Information Systems & Operations Management, volume 10, number 2, pp.327-335, December 2016.

[7] The USC-SIPI Image Database, USC University of California, Available online, retrieved from: http:// sipi.usc.edu/ database/ database.php? volume=misc, Accessed at: 30 May 2017.

[8] BZIP2 Homepage, Retrieved from: http:// www.bzip.org/ index.html, Accessed at: 30 May 2017.

[9] M. Burrows, D.J.Wheeler, A Block-sorting Lossless Data Compression Algorithm, Digital Systems Research Center, Research Report 124, May 1994.