

Near Lossless Image Compression Using Block Division Byte Compression and Block Optimization

Debashis Chakraborty, Shouvik Saha and Tanay Mukherjee

Abstract Although lossless techniques should be preferred over lossy techniques it is not always the case. This is because lossy techniques tend to decrease the overall computational time. The method described below proposes a near lossless technique that works on the spatial domain and utilizes a few properties of pixel values to reduce significantly the bit representation size of the pixel values. This algorithm utilizes a new difference concept by dividing the pixel values into groups and using a fixed value for each block to be used as a difference factor. This algorithm also approaches compression by merging the red, green, and blue components into a single component with one third the size. The algorithm provides a level of compression that remains stable irrespective of the size or dimensions of the image while leaving an acceptable range of (30–50) PSNR that is also stable.

Keywords Pixel values · Block division · Byte compression · Block optimization · PSNR

1 Introduction

The technique by which redundant data are filtered out of an image and a lot of unnecessary space is saved in a limited amount of available storage is known as image compression. There are two common forms of image compression: lossy image compression and lossless image compression [1–6]. The lossless techniques of image compression are used in situation where even a small amount of noise

Debashis Chakraborty (✉) · Shouvik Saha · Tanay Mukherjee
Department of CSE, St. Thomas' College of Engineering and Technology,
Kolkata 700023, India
e-mail: sunnydeba@gmail.com

Shouvik Saha
e-mail: sahajoy90@gmail.com

Tanay Mukherjee
e-mail: tanay.online@hotmail.com

cannot be tolerated and the exact information without any disruption is needed. Lossy techniques are more commonly used as they do not require the precision of the lossless techniques but work nearly as well. The values of the adjacent pixels of an image do not vary in a large manner so we can coordinate these into a single value. If the image is separated into different spatial blocks according to the pixel values the loss due to the compaction of the adjacent pixel can also be reduced. A novel compression/decompression algorithm is depicted in this paper, which exploits the pixel domain of an image and reduces the redundancy by working in small domains and preserving them. The proposed method provides us with advantages such as fast encoding, high ratio of compression, and simple encoding.

2 Algorithm Strategy

The image is loaded into a matrix of the same size. After loading the image the pixels are divided into 32 different groups according to their values. The groups are found by dividing the pixel values by 8 and the remainders form the new pixel values. The range of these groups or blocks are 0–7, 8–15, ..., 248–255. As seen from the division of blocks, the max value for any new pixel is 7. Hence, each of the red, green, and blue values is turned into 3-bit values at the most.

An example of this approach is shown in Fig. 1.

Byte compression is applied on these values keeping the red and green values intact and reducing the blue value into 2 bits, thus the 3-byte pixel data are reduced to a single byte. The block compression technique is further applied using 4×4 blocks for achieving a better compression.

The decompression is done in the opposite method. At first block decompression is applied regenerating the image matrix to its original size before the compression

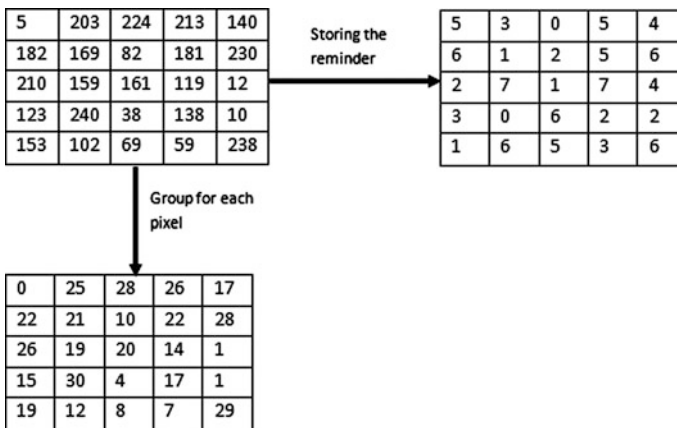


Fig. 1 Block compression.

is applied. The red, green, and blue are extracted from the bytes and associated with their representative blocks, that is, 0–31 to get the 8-bit values of red, green, and blue for each pixel thus generating the decompressed image.

3 Byte Compression/Decompression

Byte compression [7, 8] is a technique that reduces the space of a color image by storing the 3-byte RGB components into 2 bytes. The red component is reduced to 5 bits by left shifting by 3 places; the green component is also treated accordingly and the blue component is reduced to 6 bits. Thus the total size of the RGB components is reduced to 16 bits from the previous size of 24 bits. The technique used here is similar but somewhat different. Here the red component and the green component are of 3 bits each. The blue component is left shifted by one bit. This 3-bit red component, 3-bit green component, and 2-bit blue component are merged to form a single 8-bit value that is substituted in place of the original RGB values. Thus the same data that were previously stored in 3 bytes are now stored in 1 byte.

The strategy taken is shown in Fig. 2.

4 Block Compression/Decompression

In this technique the image is thought of as a cluster or collection of various $n \times n$ blocks. These $n \times n$ blocks, which are also known as masks, are replaced with their mean of average value to get a compressed and smaller image. The value of n is the main determinant in this type of compression with a higher value of n resulting in better compression but affecting the value of PSNR in a negative way. In the time of decompression each value of the compressed image matrix forms an $n \times n$ matrix of its own while repeating the same value. This way the decompression algorithm generates the decompressed image with the original dimensions [7, 9, 10].

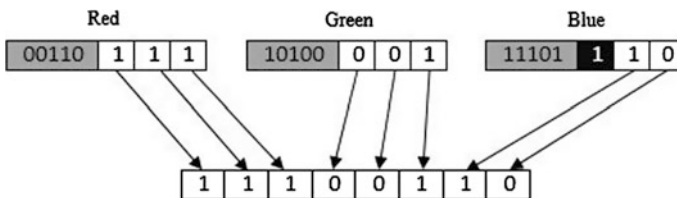


Fig. 2 Byte compression.

5 PSNR

PSNR [3–5] is commonly calculated with the help of mean squared error (MSE). MSE is defined as

$$\text{MSE} = \frac{1}{m * n} \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} [I(i,j) - K(i,j)]^2 \quad (1)$$

The PSNR (in dB) is defined as:

$$\begin{aligned} \text{PSNR} &= 10 * \log_{10} \left(\frac{\text{MAX}_i^2}{\text{MSE}} \right) \\ &= 20 * \log_{10} \left(\frac{\text{MAX}_i}{\sqrt{\text{MSE}}} \right) \\ &= 20 * \log_{10}(\text{MAX}_I) - 10 * \log_{10}(\text{MSE}) \end{aligned} \quad (2)$$

where MAX is the maximum intensity value present in the image, mostly equal to 255 for grayscale images, and I denotes a single color image on a neutral background with dimensions $m \times n$ and K denotes I with noise added to it.

The compression of an image is calculated as

$$\text{Compression \%} = ((\text{actual file size} - \text{compressed file size}) / \text{actual file size}) * 100 \quad (3)$$

6 Proposed Algorithm

The algorithm takes any colored image as input. This algorithm provides a very high compression ratio by compressing the image into a really small size.

Algorithm for Encoding

Step 1: Read the given image.

Step 2: Minimize the pixel value.

- Divide the image into groups by dividing each pixel value by 8.
- For each pixel value divided by 8 find the remainder and store it in place of the old pixel value.

Step 3: Byte compression.

- Left shift the blue component to get it into 2 bits.
- Merge the red, blue, and green components into a single 8-bit value and store it in place of the three components.

Step 4: Block compression using 4×4 blocks.

- Take a 4×4 block from the matrix.
- Calculate the average of these 16 values.
- Replace the entire block with this average value.

Algorithm for Decoding

Step 1: Read the compressed image.

Step 2: Block decompression.

- Read a pixel from the image.
- Create a 4×4 block with each value being the same as that of the pixel read.
- Replace the single pixel with this 4×4 block.

Step 3: Byte decompression.

- Read a single value from the new matrix.
- Convert the value into binary.
- Divide the binary value into three parts with bits 1–3 being the first part, bits 4–6 being the second part, and the remainder as the third part.
- Convert the first part into decimal and store as the new red component, the second part into decimal and store as the new green component, and the third part into decimal and store as the new blue component.

Step 4: Extraction of the actual pixel values.

- Take a pixel value and check to which group it belongs.
- Multiply the group number by 8 and add the pixel value to it.
- Store this value as the decompressed pixel value.
- Continue this for all components.

7 Experimental Results

This proposed algorithm for compression/decompression of images is implemented using MATLAB[®]. The images for input are taken as .ppm images. The testing is done using images of different dimensions and sizes giving us an idea of the robustness of this algorithm. Table 1 shows the comparison between the resulting image on decompression by the proposed algorithm, the image in its original quality, and the image in .jpeg format. Table 2 is a tabulated structure to compare the compression ratio between the proposed algorithm and the JPEG algorithm. The ideal PSNR ranges from 30 to 50 dB; the PSNR of the decompressed images is also calculated and shown in Table 2 to see whether they fall in the given range.

Table 1 Comparison of decompressed images

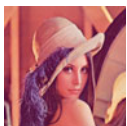



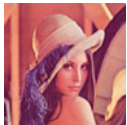







	Lenna.ppm	Mandrill.ppm	Peppers.ppm	Barbara.ppm
Image in its original quality				
Decompressed image by proposed method				
Decompressed image by JPEG algorithm				

Table 2 Comparison of compression ratios with PSNR of images

Original image	Original image size (in kB)	Compressed size by proposed algorithm (% compression)	Compressed size by JPEG algorithm (% compression)	PSNR by proposed algorithm (in dB)
Lenna.ppm	11,342	64.0 kB (99.43 %)	129 kB (98.86 %)	34.31
Mandrill.ppm	4450	25.0 kB (99.44 %)	59.0 kB (98.67 %)	34.39
Peppers.ppm	6609	39.0 kB (99.41 %)	86.9 kB (98.68 %)	34.45
Barbara.ppm	2785	16.0 kB (99.42 %)	50.2 kB (98.20 %)	34.36
Castle.ppm	2475	14.0 kB (99.43 %)	33.9 kB (98.63 %)	34.25
Satellite.ppm	1228	7.82 kB (99.36 %)	47.7 kB (96.12 %)	34.46
Parrot.ppm	4794	29.3 kB (99.39 %)	89 kB (98.14 %)	34.35
Lemon.ppm	6750	42.7 kB (99.37 %)	116 kB (98.28 %)	34.34

8 Conclusion

The proposed algorithm is a robust and effective compression algorithm for digital images. Although being lossy in nature it shows its effectiveness as it provides simple implementation, and a higher compression ratio than JPEG images in most

cases. This algorithm is applicable to color images of different dimensions and sizes giving it a much larger domain in which it can be used. With the recent trend of various multimedia applications and cross-platform media exchange, this algorithm can prove to be really useful to the ever-expanding universe of Internet and media exchange.

References

1. Salomon, D.: Data compression the complete reference, 3rd edn. Springer Press, Heidelberg
2. Saffor, A., Ng, K.-H., Ramli, A.R.: A comparative study of image compression between Jpeg and wavelet. *Malaysian J. Comput. Sci.* **14**(1), 39–45 (2001)
3. Pennebaker, W., Mitchell, J.: JPEG-still image data compression standards. Van Nostrand Reinhold (1993)
4. Padmaja, G.M., Nirupama, P.: Analysis of various image compression techniques. *ARN J. Sci. Technol.* **2**(4) (2012)
5. Acharya, T., Tsai, P.-S.: JPEG2000 Standard for Image Compression
6. http://en.wikipedia.org/wiki/Lossy_compression
7. Banerjee, A., Halder, A.: An efficient dynamic image compression algorithm based on block optimization, byte compression and run-length encoding along Y-axis. In: International Conference on Science and Information Technology (ICCSIT), (2010)
8. Ghosh, A., Chakraborty, D.: A lossy image compression by shrinking of repeating intensities in alternate dimensions and reducing bits for storage. In: ERCICA-2014, Bangalore, India, pp. 415–420
9. Halder, A, Dey, S., Mukherjee, S., Banerjee, A.: An efficient image compression algorithm based on block optimization and byte compression. In: ICISA-2010, Chennai, Tamilnadu, India (2010)
10. Das, N., Chakraborty, D.: An efficient compression for almost dual colored image using K-means clustering and block compression. In: ERCICA-2014, Bangalore, India, pp. 410–414

Authors Biography



Mr. Debashis Chakraborty is an Assistant Professor in the Department of Computer Science and Engineering, St. Thomas' College of Engineering and Technology, Kolkata, West Bengal, India. He has authored or coauthored about 17 conference papers in the area of data and image compression.



Mr. Shouvik Saha is pursuing a B. Tech degree in Computer Science and Engineering from St. Thomas' College of Engineering and Technology, Kolkata, West Bengal, India. This paper is part of the final-year project on data and image compression under the guidance of Mr. Chakraborty.



Mr. Tanay Mukherjee is pursuing a B. Tech degree in Computer Science and Engineering from St. Thomas' College of Engineering and Technology, Kolkata, West Bengal, India. This paper is part of the final-year project on data and image compression under the guidance of Mr. Chakraborty.