Journal of the Nigerian Association of Mathematical Physics
***Volume 60* (April – June *2021 Issue*), pp167 –170**
© J. of NAMP

ADAPTIVE IMAGE COMPRESSION USING THE JPEG IMAGE COMPRESSION ALGORITHM

***B. P. Ndahi*[1]*, A. Isa*[1] *and H. A. Amshi*[2]**

[1]Department of Mathematical sciences, Faculty of Science, University of Maiduguri, Maiduguri, Nigeria
[2]Department of Mathematics and Computer Science, Federal University, Gashuwa, Nigeria

Abstract

In most image processing applications, the users are only interested in parts of the information conveyed through the image. In this work, we present an adaptive technique of compressing still images. It consists first on automatically detecting faces in the image. Based on this, facial regions are given highest compression values and as one recedes to the background the compression quality keeps diminishing. Performance of this technique in terms of compression rate is compared and analyzed on images with single faces and multiple faces.

Keywords: adaptive compression, JPEG, Viola-Jones algorithm, digital images, OpenCV.

1.0. Introduction

Images in digital format often occupy a lot of space for storage when compared to text files. To transmit this kind of images requires a wide band-width. When working and processing with these kinds of images, users are usually often interested in certain regions in the image. This makes it interesting to preserve regions interesting to the users such as faces while processing.

Image compression is handled via various approaches. The Joint Photographic Experts Group (JPEG) [1] is the first international compression standard for continuous-tone still images, both greyscale and color; it happens to be generic and supports a wide variety of applications. Jpeg has very high compression ratio, and is supported in PostScript language for printing systems. Joint Photographic Experts Group 2000 (JPEG 2000) [2,3] as a compression algorithm is more efficient than Jpeg; the high demand for such technology for internet use led to its conception. It has a lossless compression mode, which is convenient for pre-printing preparation. Its data compression is complex computationally. When compressed below a certain threshold, the wavelet-based algorithm produces less block boundary artifacts, but slightly less detailed images compared to conventional JPEG compression. The Run-length encoding also known as RLE [4] is computationally basic. Its compression works well with files that contain a lot of repetitive data such as text files if they contain a lot of space of the same color. With RLE, computer color images (for example, architectural drawings) can also be highly compressed. Lempel–Ziv–Welch (LZW) compression [5] works best for monochrome images and text files that contain a lot of repetitive data. Compressed files that do not contain any duplicate pattern or information may grow larger than the original files; LZW compression is simple computationally; it is a compression technique that has been around for quite some time and is fairly old. New computer systems usually implement more efficient algorithms. Huffman compression algorithm [6] is one of the oldest compression algorithms but still rated as one of the most powerful data compression algorithms. It can be classified as a lossless data compression algorithm because no bits are lost during the compression or decompression process. It is based on coding technique using symbol wise method. The algorithm is one of the algorithms used to compress text. Huffman coding is based on the frequency of occurrence of a data item like pixel in images. The technique uses a lower number of bits to encode the data in to binary codes that occurs more frequently. It is implemented in JPEG files. All these systems handle compression but none by default allowed us to adaptively compress facial regions with high quality then other areas with lower and lower compression quality as we recede from the facial area to the background. To mitigate this problem, we proposed to detect the faces using Viola-Jones algorithm, then created a unique algorithm to find compression levels of blocks and based on this, the output is encoded to jpeg based on assigned compression levels of blocks.

Section 2 of this paper gives details on the suggested principle of finding faces. Section 3 describes an adaptive still image compression approach. Experimental results were presented in section 4. Finally, we concluded the paper in section 5.

Corresponding Author**:** Ndahi B.P., Email: ndahibarka@yahoo.com, Tel: +2348120262537

2.0. Region of interest detection

The first stage of the proposed strategy consists of detecting and drawing rectangles around faces in the image. The detection can be done via varied means but in this work, it was carried out via Viola-Jones algorithm [7,8,9] implemented via Open Source Computer Vision Library (OpenCV) [10,11,12] functions on Visual C++. In this manner, faces are found which are the objects of interest in the image.

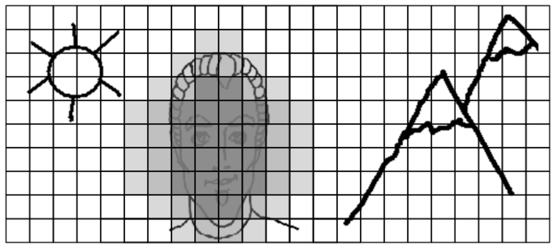


Fig. 1. Matrix of the levels of compressions for the image where the darkest spots represent areas with highest levels of compression quality.

3.0. Adaptive compression

The adaptive compression implemented is based on detecting faces as explained above, determining the degree of compression of blocks then finally encoding into Jpeg. Fig. 1 represents a diagram highlighting the suggested method.

3.1. Region of interest and compression

The aim of the proposed approach is to sustain the information transmitted through the image in the rectangles drawn around detected faces with the highest possible value while the background gets lower compression quality, with slight transitions from high to low.

3.2. Determining the degree of compression of blocks

For this step, it was necessary to write a personal algorithm.

The function returns a vector of integer values, the values vary in the range [0-100] in accordance with the required compression quality. A value of 100 is the best compression quality.

Determination of the compression ratio for each block of the image is performed based on its proximity to the person's face. In this case, the following algorithm is used (based on the final file size):

The algorithm works in the following manner:

–based on the list of tilted rectangles [13] received from the human face detection module, the number of blocks that make up the face areas in the image is calculated;

–based on the final file size, the compression rates are calculated for the areas of the person's face and areas of the rest of the background;

–to reduce the sharpness of the transition between the areas of faces to be compressed with high quality and the areas of the background, in a certain neighborhood of faces for the corresponding blocks, the degree of compression is gradually reduced from the degree of compression of the face area to the degree of compression of the background areas.

3.2.1. Description of the algorithm for calculating the compression level of each 8x16 blocks

An image is transmitted at the input of the algorithm.

First, the algorithm must find the center of the faces of all recognized people in the image using OpenCV functions and libraries [14]; then store their coordinates in vector (array) using this formula:

$$\mathrm{centerX_i} = \mathrm{faceX_i} + \frac{\mathrm{faceWidth_i}}{2}, \qquad (1)$$

$$\mathrm{centerY_i} = \mathrm{faceY_i} + \frac{\mathrm{faceHeight_i}}{2}, \qquad (2)$$

where $\mathrm{centerX_i}$ is x coordinate of the center of the ith face, $\mathrm{centerY_i}$ is y coordinate of the center of the ith face, $\mathrm{faceX_i}$ is x coordinate from the list of tilted rectangles received from the human face detection module, $\mathrm{faceWidth_i}$ is width from the list of tilted rectangles received from the human face detection module, $\mathrm{faceY_i}$ is y coordinate from the list of tilted rectangles received from the human face detection module, $\mathrm{faceHeight_i}$ is height from the list of tilted rectangles received from the human face detection module.

Then, it is necessary to generate a list of compression ratios for all Jpeg Minimum Coded Unit (MCU) blocks (usually 8x16 pixels) [1] of which the image consists.
While looping through image rows by a counter equivalent to MCU height and looping through image columns by a counter equivalent to MCU width:
If the sum of the counter and the MCU block height exceeds the width of the image ($i + MCU\_H \geq image\_rows$) then it divides the new block by Eq. (3) else by Eq. (4):

$$X = i + \frac{(image\_rows - i)}{2}, \quad (3)$$

$$X = i + \frac{MCU\_H}{2}, \quad (4)$$

where X is x coordinate of the new block, image_rows is image width, MCU_H is block height of MCU, i is counter.
If the sum of the counter and the block width exceeds the image height ($j + MCU\_W \geq image\_cols$) then it divides the new block by Eq. (5) else by Eq. (6):

$$Y = j + \frac{(image\_cols - j)}{2}, \quad (5)$$

$$Y = j + \frac{MCU\_W}{2}, \quad (6)$$

where Y is the y coordinate of the new MCU block, image_cols is image height, MCU_W is block width, j is counter.
After that, when there are many faces in the image, it is necessary to find the minimum distance to select for compression level. That is, the minimum distance from any given MCU block to center of faces in the image. The minimum distance is calculated by the formula:

$$dist = \sqrt{((X - X_k)^2 + (Y - Y_k)^2)}, \quad (7)$$

where dist is the distance, X and Y are X and Y coordinates of the current face, X_k and Y_k are the X_k th and Y_k th coordinates of the face in the vector (array).
After cycling through the vector containing minimum distances from MCU blocks to center of all faces, the degree of compression of all the blocks is calculated one after the other in relationship to the maximum distance in the vector distances by the formula:

$$scl = \frac{distances_i}{max}, \quad (8)$$

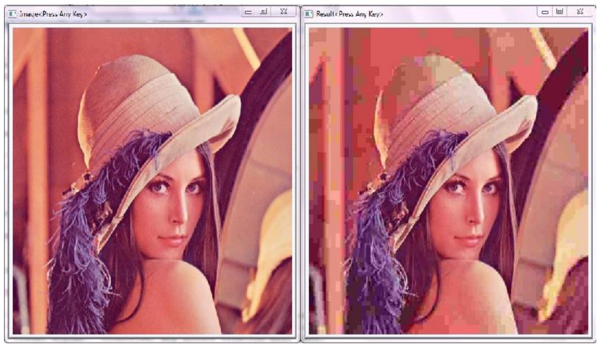
where scl is compression level for the MCU block, $distances_i$ is vector (array) having all the most minimum distances from MCU blocks to center of all faces, max is the most biggest value in the vector (array) $distances_i$.

3.3. Image coding to jpeg

The output is encoded to jpeg using jpeg image compression algorithm [15,16,17] based on assigned compression levels of blocks.

4.0. Results and interpretation

The method was applied to images with one or several faces. Fig. 2 presents the results of the application of the method on an image with a single face.



(a) (b)

Fig. 2. (a) Original Lena image (single face): File size = 83kb. File type=JPEG (b) Compressed image: File size: 41kb. File size reduction= 51%. File type=JPEG.

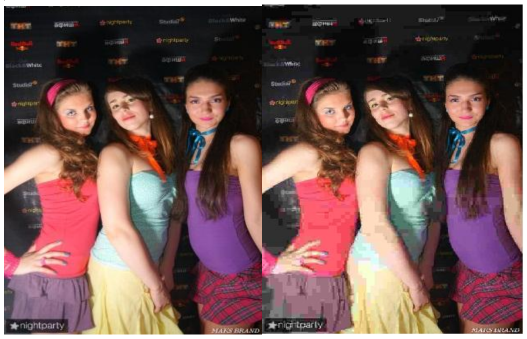


Fig. 3. (a) Original Girls image (multiple faces): File size = 86kb. File type=JPEG (b) Compressed image: File size: 60kb. File size reduction = 30%. File type=JPEG.

Fig. 3 shows the result of applying the method on image with several faces. The starting images are of type Jpeg. We applied compression to the image and tested the method with various images. Table 1 shows the result when adaptive compression is applied to the images.

Table 1: Adaptive compression results

	Original size	Compressed size	File size reduction
Lena (single face)	83kb	41kb	51%
Girls (multiple faces)	86kb	60kb	30%

Comparison of these results reveals that adaptive compression makes it possible to reach high compression ratios while retaining high visual quality, especially in the facial regions. These results depend on the dimension of the faces, and certainly the number of faces existing in the image.

According to the results of the work, it was possible to achieve roughly an average of 30% usually when there are many faces (and 50% if there is a single face) adaptive compression degree while maintaining the visual quality of the image.

5.0. Conclusion

In this work, we presented an adaptive compression technique based on the fact that users are interested only in part of the information conveyed through the image. The faces are detected using an automatic method based on Viola-Jones algorithm. The whole image is then compressed in such a way that the facial area has highest compression value with the background having lowest and there is a slight transition from region of high quality to that of low quality then the whole thing is encoded into jpeg. The results obtained showed that this technique makes it possible to obtain a high compression ratio while preserving good visual image quality especially in the facial areas.

References

[1] G. K. Wallace, “The JPEG still picture compression standard”, IEEE Transactions on Consumer Electronics, Vol. 38, No. 1, 1992, pp. xviii–xxxiv, https://doi.org/10.1109/30.125072

[2] A. Skodras, C. Christopoulos and T. Ebrahimi, “The jpeg 2000 still image compression standard”, IEEE Signal processing magazine, vol. 18, No. 5, 2001, pp. 36-58.

[3] L. Leurs, “JPEG2000 compression”, https://www.prepressure.com/library/compression-algorithm/jpeg2000, Accessed March 2021

[4] S. K. Bandyopadhyay, T. U. Paul and A. Raychoudhury, “Image compression using approximate matching and run length”, International Journal of advanced Computer science and applications, vol. 2, No. 6, 2011, pp. 117-121.

[5] H. N. Dheemanth, “LZW Data Compression”, American Journal of Engineering Research, vol. 3, No. 2, 2014, pp. 22-26.

[6] Y. S. Triana and A. Retnowardhani, “Blowfish algorithm and Huffman compression for data security application”, In IOP Conference Series: Materials Science and Engineering , vol. 453, No. 1, 2018, p. 012074.

[7] R. Gupta, “Breaking Down Facial Recognition: The Viola-Jones Algorithm”, 2019, https://towardsdatascience.com/the-intuition-behind-facial-detection-the-viola-jones-algorithm-29d9106b6999

[8] L. Shi and J. H. Lv, “Face detection system based on AdaBoost algorithm”, Applied Mechanics and Materials, vol. 380, 2013, pp. 3917-3920.

[9] P. Viola and M. J. Jones, “Rapid Object Detection using a Boosted Cascade of Simple Features”, IEEE CVPR, 001, pp. 1-9.

[10] P. Y. Kumbhar, M. Attaullah, S. Dhere and S. Hipparagi, “Real time face detection and tracking using OpenCV”, Int. J. Res. Emerg. Sci. Technol, vol. 4, issue 4, 2017, pp. 39-43.

[11] K. Pulli, A. Baksheev, K. Kornyakov and V. Eruhimov, “Real-time computer vision with OpenCV”, Communications of the ACM, Vol. 55, Issue 6, 2012, pp. 61-69.

[12] G. Bradski and A. Kaehler, “OpenCV”, Dr. Dobb’s journal of software tools, 3, 2000, pp. 1-81.

[13] R. Lienhart and J. Maydt, "An extended set of haar-like features for rapid object detection", In Proceedings. international conference on image processing, Vol. 1, 2002, pp. I-I.

[14] OpenCV Reference Manual, v 2.1, March 18, 2010, pp. 1-1104

[15] M. S. AL-Ani and F. H. Awad, “The JPEG Image Compression Algorithm”, International Journal of Advances in Engineering & Technology, vol. 6, No. 3, 2013, pp. 1055-1062, http://www.e-ijaet.org/media/2I15-JAET0715522_v6_iss3_1055to1062.pdf

[16] A. Townsend, “JPEG: Image compression algorithm”, 2017, http://pi.math.cornell.edu/~web6140/TopTenAlgorithms/JPEG.html

[17] M. Abuzaher and J. Al-Azzeh, “JPEG Based Compression Algorithm”, International Journal of Engineering and Applied Sciences (IJEAS), vol. 4, No. 4, 2020, pp. 95-97.