Bulletin of Electrical Engineering and Informatics
Vol. 10, No. 5, October 2021, pp. 2607∼2616
ISSN: 2302-9285, DOI: 10.11591/eei.v10i5.2563

A spatial image compression algorithm based on run length encoding

Abdel Rahman Idrais[1], **Inad Aljarrah**[2], **Osama Al-Khaleel**[3]
[1,2,3]Department of Computer Engineering, Jordan University of Science and Technology, Irbid, Jordan
[1]Department of Electrical and Computer Engineering, Concordia University, Montreal, Canada

Article Info

Article history:

Received Apr 22, 2020
Revised Apr 29, 2021
Accepted Jul 29, 2021

Keywords:

Artificial intelligence
Compression algorithms
Image coding
Inter-pixel redundancy
Run-length coding

ABSTRACT

Image compression is vital for many areas such as communication and storage of data that is rapidly growing nowadays. In this paper, a spatial lossy compression algorithm for gray scale images is presented. It exploits the inter-pixel and the psycho-visual data redundancies in images. The proposed technique finds paths of connected pixels that fluctuate in value within some small threshold. The path is calculated by looking at the 4-neighbors of a pixel then choosing the best one based on two conditions; the first is that the selected pixel must not be included in another path and the second is that the difference between the first pixel in the path and the selected pixel is within the specified threshold value. A path starts with a given pixel and consists of the locations of the subsequently selected pixels. Run-length encoding scheme is applied on paths to harvest the inter-pixel redundancy. After applying the proposed algorithm on several test images, a promising quality vs. compression ratio results have been achieved.

Corresponding Author:

Inad Aljarrah
Department of Computer Engineering
Jordan University of Science and Technology
Irbid, Jordan
Email: inad@just.edu.jo

1. INTRODUCTION

With the vast increase of usage of social media among people, the amount of images and videos transmitted is increasing dramatically [1], [2]. For example, on January 2019; over 240 billion photos had been uploaded by Facebook website users, with 350 million new daily photos. This requires a huge amount of storage and a very high-speed communication link [3]. Therefore, a scheme to reduce the size of images while maintaining an acceptable image resolution and quality is needed [4]. Image compression is a technique that reduces the size of an image by minimizing the number of bits needed to represent it [5]-[8]. Image compression methods evaluation is an application dependent. A good image compression technique for a given application is not necessarily good for another one [9]-[11].

There are two main categories of image compression techniques; lossless and lossy. In lossless image compression, the quality of the compressed image is similar to the quality of the original image. All information of the original image are preserved in the compressed one [12]. On the other hand, lossy image compression slightly alters the information of original image in a way that decreases the quality of the compressed image comparing to the original one [13]-[15]. In order to judge the efficiency of a compression scheme, a performance measurement is required [16]. The most commonly used performance measurements are: the compression ratio, the execution time of the compression and decompression processes, and the signal to noise ratio [17]-[19]. The compression ratio (CR), which is calculated using (1), is defined as the ratio between the size of the original image (N_1) and the size of the compressed image (N_2).

$$CR = \frac{N_1}{N_2} \tag{1}$$

High speed compression and decompression are necessary for real time applications. In lossy approaches, the peak signal to noise ratio (PSNR) measures the quality of the decompressed image [20]. The value of the PSNR in lossless techniques is useless as the original and the decompressed images are identical. The PSNR is calculated using (2) and (3).

$$PSNR = 10 \log_{10} \frac{255^2}{RMSE^2} \tag{2}$$

$$RMSE = \sqrt{\frac{1}{MN} \sum \sum ((f(x,y) - f'(x,y)))^2)} \tag{3}$$

where (RMSE) is the root mean square error, M and N represent the length and width of image respectively, $f(x,y)$ is the pixel value in the original image at x and y coordinates, and $f'(x,y)$ is the pixel value in the decompressed image at x and y coordinates.

Every image contains some redundant data that can be exploited in image compression [21], [22]. By eliminating such redundant data, the amount of data required to represent the image is reduced. Three different types of redundancy can be found in an image: inter-pixel redundancy, psycho-visual redundancy, and coding redundancy. Inter-pixel redundancy means that the intensities of neighboring pixels are not statistically independent and a correlation do exist among them. Inter-pixel redundancy (also called the spatial redundancy) is related to the fact that a pixel value can be predicted from its location and the values of its neighboring pixels.

The human eye does not respond to all incoming visual information with equal vulnerability; some information has less relative importance. This property is called psycho-visual redundancy and can be used to remove some redundant information without affecting how the human eye perceive the image. Most of image compression algorithms exploit the psycho-visual redundancy, like the discrete cosine transform (DCT) [23]-[25].

Different image compression schemes have been proposed in literature. These schemes are either lossy or lossless. In either case, the ultimate goal is to achieve high compression ratio with reasonable execution time. Some of these schemes are summarized in this section. A review of the compression techniques in digital image processing and a brief description of the main technologies and traditional format that commonly used in image compression have been presented in [26]. Moreover, the formats that are used to reduce redundant information in an image are presented.

The work in [27] presents a new method to find the most redundant image structure elements; like patterns, textures, and edges to compress and encode them. This method is called super-spatial prediction. The goal of super-spatial prediction method is to recognize the best prediction of structure elements by using the idea of motion prediction in video coding. To find the prediction of current image block the super-spatial prediction scheme search for best prediction by calculating the minimum difference between the blocks.

Khalil the authors of [28] introduce a new method of run-length coding; the new method is easier and more efficient than the standard one. It computes the discrete cosine transform (DCT) and works on its quantized coefficients. The experimental results of Khalil enhance the compression ratio compared with the older version of run-length coding method. Because of the speed and the simplicity of his method; it may be used in data compressors such as video compression.

A new compression scheme based on the pixels' locations is proposed in [29]. The scheme simply divides the image into 4×4 non-overlapping blocks and works on each block separately. It takes the most frequent pixel in that block and deletes all of its occurrences, and does that for the other less frequent pixels with the same way. They compare their scheme with known schemes like GIF, TIFF, and PNG. When applying them on 200 textbook images, they claim gaining better compression ratio in 83% of the cases.

Adaptive thinning algorithms by using recursive point removal methods is a new concept for compression that has been proposed by [30]. This scheme is more appropriate for modeling sharp edges and related features in digital images. Their results are compared with another compression method called SPIHT, and the quality is represented by the peak signal to noise ratio (PSNR). Their scheme is tested on some sample images and got better results than SPIHT at the sharp edges in the decompressed image.

Azman authors in [31] have combined predictive differential pulse code modulation (DPCM) and integer wavelet transform (IWT) in order to achieve a hybrid prediction lossless image compression approach. They have analyzed the performance of their proposed algorithm by calculating the entropy and the compression ratio. Experimentally, sequence of DPCM-IWT-huffman gives the best compression results.

The work of [32] presents two lossless compression schemes, the min-max differential (MMD) and the min-max predictive. The two methods are combined with the existing compression methods to enhance them. The authors apply the proposed schemes for medical images. They test some images of CT brain scans and the results show an improvement over other compression methods such as Huffman encoding, arithmetic coding, and Lempel-Zif.

Two algorithms are proposed by [33] to maximize the performance of JPEG by combining the optimization of discrete cosine transform, huffman code, and quantization of JPEG encoder. The goal of first algorithm is to find the optimal discrete cosine transform indices in the form of run-size pairs. Then, the second algorithm iteratively optimized run-length coding, quantization, and Huffman coding. Both of the proposed algorithms can be used in digital camera image compression, transcoding, and MPEG frame optimization.

A new predictive lossless image compression scheme is found in [34]. The proposed algorithm uses the gradient edge detector as a predictor to remove the spatial data redundancy. The performance of the proposed algorithm is efficient for 12-bit medical images and has a good compression ratio versus JPEG-LS.

A novel medical image compression approach has been proposed in [35]. The approach combines geometric active contour model and quincunx wavelet transform. A level set for an optimal reduction is has been used to localize all the part that contain the pathological. The quincunx wavelet coupled is then used with the set partitioning in hierarchical trees (SPIHT) algorithm. A satisfactory results have been achieved.

The rest of this paper is organized is being as: section 2 discusses the proposed compression technique and illustrates it using an example. The experimental results are presented and discussed in section 3. Finally, the conclusions are provided in section 4.

2. RESEARCH METHOD

The proposed algorithms exploit the psycho-visual and inter-pixel data redundancies. They are lossy compression algorithms that work with pixels' values in the spatial domain. Thus, the decompressed image will not be exactly similar to the original image.

The proposed method finds paths between neighboring pixels that have same values within some threshold. The path is calculated by investigating the 4-neighbors of a pixel, as shown in Figure 1 (a). One of the neighbors of the central pixel is chosen according to a certain technique. The location of the selected pixel with respect to the central pixel is stored in the path and marked as visited. The locations are relative to the middle pixel. A location can be UP, DOWN, LEFT, or RIGHT as illustrated in Figure 1 (b). A path consists of a starting pixel value and continues with a sequence of pixels locations that are subsequently selected. The criterion to select a pixel out of 4-neighbors is based on two conditions: the selected pixel must be unvisited and the difference between the starting pixel value and the selected pixel value is within a threshold. When there are two or more pixels, among the 4-neighboring pixels, satisfy the conditions, the selection is performed based on the values of the neighbors of those neighbors. In such case, the pixel that has the minimum difference with one of its neighbors is selected. A path ends once none of the 4-neighbors of a selected pixel satisfies the condition. The image is investigated from top-left moving down-right column by column.

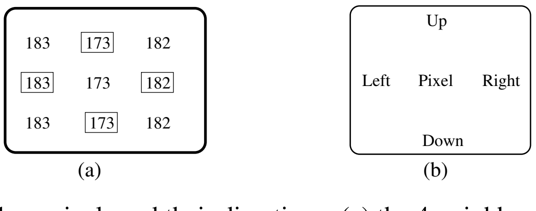


Figure 1. The 4-neighbors pixels and their directions: (a) the 4-neighbors of a centered pixel, (b) the directions of 4-neighbors pixels

In the compression stage, the compressed image consists of a series of connected paths. Each path is represented using the value of the starting pixel followed by the consecutive pixels' locations (UP, DOWN, LEFT, or RIGHT). Finally, a STOP code is inserted at the end to indicate the termination of the path. One example of a path is shown in Figure 2. The number of bits needed to represent the value of the starting pixel is 8. The four directions that are used to specify the location of a pixel can be replaced by three path directions (LEFT, RIGHT, and STRAIGHT), because the previous pixel will not be chosen again. Also, it should be noted that a right location will be added at the beginning of any path so that when the path directions are calculated the length of the path is preserved due to the fact that the number of directions will be one less than the number of locations.

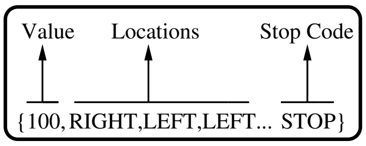


Figure 2. A sample path which contains: value of starting pixel, the following pixels directions and a stop code

Table 1 shows how to map the location of a pixel by the equivalent path direction. It should be pointed out that the STOP code can be represented using three consecutive LEFT or RIGHT direction codes. This can be done because such a sequence is impossible to occur in a real path. This leads to having any path being represented using only the path directions (LEFT, RIGHT, and STRAIGHT) and the starting 8-bit pixel value.

Table 1. Conversion from pixel location to path direction and from path direction to pixel location

Conversion from pixel location to path direction		
Location		2* Path direction
Current pixel	Next pixel	
4*UP	UP	STRAIGHT
	LEFT	LEFT
	RIGHT	RIGHT
	DOWN	——-
4*DOWN	UP	——-
	LEFT	RIGHT
	RIGHT	LEFT
	DOWN	STRAIGHT
4*LEFT	UP	RIGHT
	LEFT	STRAIGHT
	RIGHT	——-
	DOWN	LEFT
4*RIGHT	UP	LEFT
	LEFT	——-
	RIGHT	STRAIGHT
	DOWN	RIGHT

Conversion from path direction to pixel location		
Direction		2*Location
Current	Next	
3*STRAIGHT	LEFT	UP
	RIGHT	DOWN
	STRAIGHT	RIGHT
3*LEFT	LEFT	LEFT
	RIGHT	RIGHT
	STRAIGHT	UP
3*RIGHT	LEFT	DOWN
	RIGHT	UP
	STRAIGHT	LEFT
3*DOWN	LEFT	RIGHT
	RIGHT	LEFT
	STRAIGHT	DOWN

As an example, Figure 3 (a) shows a path of pixels (R, D, R, D, D, R, U, R, U, L, U). The path is created based on the location of next pixel with respect to current one. After adding a right at the beginning, the path locations become (R, R, D, R, D, D, R, U, R, U, L, U). All selected pixels must be within the threshold, and each pixel value will be replaced by the value of the first pixel. Figure 3 (b) shows how to construct the path using its direction (S, R, L, R, S, L, L, R, L, L, R).

In the decompression stage, we need to reconstruct the compressed image using all paths. Each path contains the value of the starting pixel, path directions (LEFT, RIGHT, and STRAIGHT), and ends with the STOP code. The directions of each path are converted to pixel locations (as shown in Table 1). Then the decompressed image is constructed by filling the value of the starting pixel on each pixel location.

There are three different approaches that will be explained in the next three sections. For selecting and choosing the most appropriate neighbors of a given pixel, three different approaches have investigated. These approaches are: compression with search depth (D=1), compression with search depth (D=2), and compression with search depth (D=5).

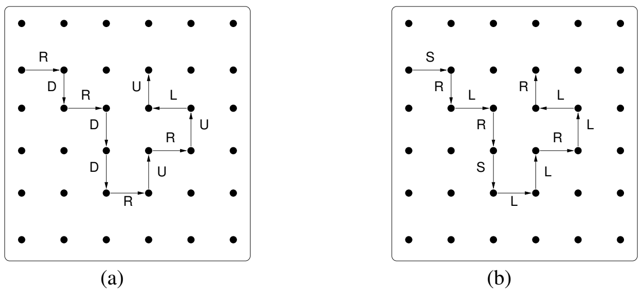


Figure 3. Example of construction the path based on pixels locations and using the path directions: (a) the path based on pixels locations, (b) the path using path directions

2.1. Compression with search depth (D=1)

In the (D=1) approach, initially, the top-left pixel value is chosen as a starting pixel and is added to the path. Then, one of the 4-neighbors (UP, DOWN, LEFT, or RIGHT) that is unvisited and has minimum difference within the threshold value is selected. The process is repeated for the selected pixel until all neighbors of the newly selected pixel are visited or none of them is within the threshold value. The Stop code is then added to the path. An unvisited pixel from top left moving down right is chosen to be a starting pixel in the second path. The pseudo-code in Figure 4 demonstrates this approach. During constructing the path, if there are two or more neighbors that have the same difference then one of them is randomly chosen. Once all pixels in the image are marked as visited, the path set is completed and is ready to be converted from path location codes to path direction codes.

```
D = 1: Path = Function (Image_Matrix, threshold)
1.  FOR each pixel in Image_Matrix
2.  IF Visited_Matrix (I, J) IS 'not visited' THEN
3.  SET Visited_Matrix (I, J) to 'visited'
4.  SET First_Value to Image_Matrix (I, J)
5.  INCREMENT Counter
6.  SET Path (Counter) to First_Value
7.  REPEAT
8.  SET UP_ Neighbor to Image_Matrix (I - 1, J)
9.  SET Left_ Neighbor to Image_Matrix (I, J - 1)
10. SET RIGHT_ Neighbor to Image_Matrix (I, J + 1)
11. SET DOWN_ Neighbor to Image_Matrix (I + 1, J)
12. IF there is at least one Neighbor un-visited AND
    difference within threshold THEN
13. CHOOSE the un-visited neighbor that generates minimum
    difference within threshold AND STORE its value IN
    Path_location
14. UPDATE I to new_I
15. UPDATE J to new_J
16. INCREMENT Counter
17. SET Path (Counter) to Path_location
18. ELSE
19. INCREMENT Counter
20. SET Path (Counter) to 'STOP'
21. END IF
22. UNTIL all pixels of Image_Matrix (I, J) are visited
23. END IF
24. END FOR
```

Figure 4. A pseudo-code which presents (D=1) approach

In Figure 4, Image_Matrix is the matrix that contains all pixel's values, Visited_Matrix is the matrix that marks each pixel as 'visited' or 'not visited', First_Value is the pixel value of the current path, path is an array that stores the pixels values and locations, threshold is the threshold value, Path_location contains one value form four pixels locations (UP, LEFT, RIGHT, and DOWN), UP_Neighbor is the upper neighbor of a pixel, Left_Neighbor is the left neighbor of a pixel, RIGHT_Neighbor is the right neighbor of a pixel, DOWN_Neighbor is the down neighbor of a pixel, I is the x Coordinate of the pixel, J is the y coordinate of the pixel, and counter is incremented when a new pixel is marked as visited.

2.2. Compression with search depth (D=2)

In search depth (D=2), the top-left pixel value is initially added to path as in search depth (D=1). However, the next pixel in the path is not directly determined based on investigating the 4-neighbors. Instead, in this case all neighbors pixels that are within the specified threshold are determined. Let assume that these pixels belongs to group A. Then the 4-neighbors of each pixel in group A are investigated and those satisfy the criteria are specified. Hence, the pixel from group A that owns more pixels that satisfy the criteria is chosen to be in the path. When all neighbors of selected pixel are visited or none of them is within the threshold value, the stop code is added to path and we select an unvisited pixel from the top left moving down right. The pseudo-code of this approach is depicted in Figure 5. After visiting all pixels in the image, the paths are complete and are ready for conversion from path location to path direction.

```
D = 2: Path = Function (Image_Matrix, threshold)
1.  FOR each pixel in Image_Matrix
2.    IF Visited_Matrix (I, J) IS 'not visited' THEN
3.      SET Visited_Matrix (I, J) to 'visited'
4.      SET First_Value to Image_Matrix (I, J)
5.      INCREMENT Counter
6.      SET Path (Counter) to First_Value
7.      REPEAT
8.        SET UP_ Neighbor to Image_Matrix (I - 1, J)
9.        SET Left_ Neighbor to Image_Matrix (I, J - 1)
10.       SET RIGHT_ Neighbor to Image_Matrix (I, J + 1)
11.       SET DOWN_ Neighbor to Image_Matrix (I + 1, J)
12.       IF there are at least two neighbor un-visited AND
          difference within threshold THEN
13.         CHOOSE a neighbor that has at least one neighbor
            within threshold AND STORE its value IN Path_location
14.         UPDATE I to new_I
15.         UPDATE J to new_J
16.         INCREMENT Counter
17.         SET Path (Counter) to Path_location
18.       ELSE IF there is one Neighbor un-visited AND
          difference within threshold THEN
19.         UPDATE I to new_I
20.         UPDATE J to new_J
21.         INCREMENT Counter
22.         SET Path (Counter) to Path_location
23.       ELSE
24.         INCREMENT Counter
25.         SET Path (Counter) to 'STOP'
26.       END IF
27.     UNTIL all pixels of Image_Matrix (I, J) are visited
28.   END IF
29. END FOR
```

Figure 5. A pseudo-code which presents (D=2) approach

2.3. Compression with search depth D=5

Similarly as in search depth (D=1) and (D=2), in this approach, the top-left pixel value is initially added to the path. The 4-neighbors of the pixel are investigated and their neighbors are investigated and the neighbors of the neighbors are investigated up to 5 levels as long as the pixels do satisfy the unvisited and threshold criteria. Therefor, this approach covers the search depth (D=3)and (D=4). The direction which has more pixels that satisfy the criteria is adopted and the pixel in that direction is added to the path. This added pixel should have up to five unvisited sub-neighbors that are within the threshold. It might have four or three sub-neighbors depending on how many levels we could span. When all neighbors of the selected pixel are visited or none of them is within threshold value, the stop code is added to path and we select an unvisited pixel from top left moving down right. The pseudo-code of this approach is depicted in Figure 6. After visiting all pixels in the image, the paths are complete and are ready to be converted from path location to path direction.

```
D = 5: Path = Function (Image_Matrix, threshold)
FOR each pixel in Image_Matrix
    IF Visited_Matrix (I, J) IS 'not visited' THEN
        SET Visited_Matrix (I, J) to 'visited'
        SET First_Value to Image_Matrix (I, J)
        INCREMENT Counter
        SET Path (Counter) to First_Value
        REPEAT
            SET UP_ Neighbor to Image_Matrix (I - 1, J)
            SET Left_ Neighbor to Image_Matrix (I, J - 1)
            SET RIGHT_ Neighbor to Image_Matrix (I, J + 1)
            SET DOWN_ Neighbor to Image_Matrix (I + 1, J)
            IF there are at least two neighbor un-visited AND difference within threshold THEN
                CHOOSE a neighbor that has up to five sub-neighbors within threshold AND STORE its value IN Path_location
                UPDATE I to new_I
                UPDATE J to new_J
                INCREMENT Counter
                SET Path (Counter) to Path_location
            ELSE IF there is one Neighbor un-visited AND difference within threshold THEN
                UPDATE I to new_I
                UPDATE J to new_J
                INCREMENT Counter
                SET Path (Counter) to Path_location
            ELSE
                INCREMENT Counter
                SET Path (Counter) to 'STOP'
            END IF
        UNTIL all pixels of Image_Matrix (I, J) are visited
    END IF
END FOR
```

Figure 6. A pseudo-code which presents (D = 5) approach

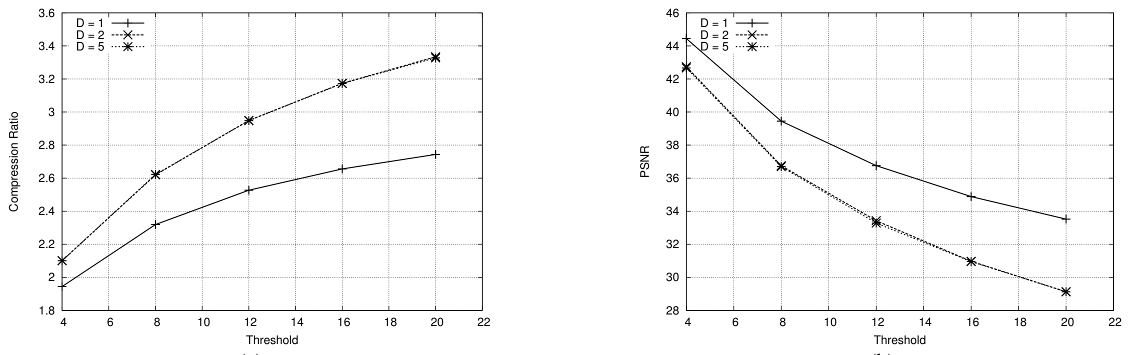
3. RESULT AND DISCUSSION

The compression techniques presented above have been implemented using Matlab. The efficiency of the algorithms has been tested using some standard images like Peppers, Lena, and Cameraman. The test is carried out by compressing and decompressing the images and then comparing the output (decompressed) image with the input (original) image. For each image, the signal to noise ratio and the root mean square error is calculated as measures for the quality of the output image. Also, the compression ratio for the compressed images are calculated. All of these results have been studied versus increasing the threshold value. The results have been collected for threshold values of 4, 8, 12, 16, and 20. All the different search depth approaches (D=1), (D=2), and (D=5) have been considered. All of the collected results have been listed in Table 2.

Table 2. Compression ratio and PSNR in all depths with threshold equal to 4, 8, 12, 16, and 20

2*Threshold	2*Image	(D = 1)		(D = 2)		(D = 5)	
		CR	PSNR	CR	PSNR	CR	PSNR
3*4	Peppers	1.89	43.33	2.01	41.84	2.00	41.84
	Lena	1.88	43.32	1.98	41.79	1.99	41.86
	Cameraman	2.27	44.34	2.40	41.78	2.39	41.68
3*8	Peppers	2.37	38.39	2.63	35.99	2.64	35.96
	Lena	2.34	38.4	2.60	35.99	2.62	36.12
	Cameraman	2.67	39.55	2.94	36.12	2.92	36.19
3*12	Peppers	2.62	35.69	3.02	32.73	3.04	32.71
	Lena	2.57	35.79	2.99	32.68	3.01	32.87
	Cameraman	2.87	37.08	3.24	32.91	3.23	32.71
3*16	Peppers	2.76	33.94	3.26	30.37	3.28	30.25
	Lena	2.70	34.12	3.23	30.35	3.25	30.54
	Cameraman	2.99	35.26	3.43	30.70	3.41	30.57
3*20	Peppers	2.84	32.71	3.43	28.67	3.44	28.46
	Lena	2.79	32.86	3.40	28.51	3.42	28.67
	Cameraman	3.06	34.03	3.54	28.34	3.53	28.13

To demonstrate the impact of increasing the threshold value on the compression ratio (CR) and on the peak signal to noise ratio (PSNR), in Figure 7 (a), the CR is plotted versus the threshold value. Also, in Figure 7 (b), the PSNR is plotted versus the threshold value. This has been done for the Lena image. It can be easily figured out that the CR increases with the increase of the threshold value as shown by Figure 7 (a). While, the quality of the image decreases as the threshold is increased as shown by Figure 7 (b).



(a) (b)

Figure 7. Threshold vs. compression ratio and PSNR for depth one, two, and five: (a) threshold vs. compression ratio, (b) threshold vs. PSNR

4. CONCLUSION

A spatial lossy compression technique that is based on inter-pixel and psych-visual redundancies has been proposed. The approach in the proposed technique is to find paths between neighboring pixels that have close values within some threshold. Paths are constructed by investigating the neighborhood of the current pixel for a certain depth. In the nearest depth, the next level neighborhood is investigate. While, in next higher depth neighborhood of the neighborhood is investigate and son. In order to determine the direction of the path, the value of the current pixel is compared with the values of all unvisited pixels in its neighborhood. This is repeated for a number of times that is basted on the chosen depth. Three different depths have been tested and the compression ratio versus the image quality results have been reported for five different threshold values. Increasing the search depth from (D=1) to (D=2) would improve the compression ration with a loss in the image quality. On the other hand, very minor changes, on the compression ration and the image quality, arise when moving from search depth (D=2) to (D=5). The techniques have been implemented using MATLAB. Promising compression results have been achieved.

REFERENCES

[1] J. Liu, S. Wang, and R. Urtasun, ”Dsic: Deep stereo image compression,” in *2019 IEEE/CVF Interna- tional Conference on Computer Vision (ICCV)*, 2019, pp. 3136–3145.

[2] J. L. Nunez and S. Jones, ”Run-length coding extensions for high performance hardware data compres- sion,” in *IEE Proceedings-Computers and Digital Techniques*, vol. 150, no. 6, pp. 387–395, 2003, doi: 10.1049/ip-cdt:20030750.

[3] D. T. Võ, S. Lertrattanapanich and Y. Kim, ”Visually lossless compression for color images with low line memory requirement using non-uniform quantizers,” *2011 IEEE International Conference on Consumer Electronics (ICCE)*, 2011, pp. 639-640, doi: 10.1109/ICCE.2011.5722783.

[4] E. Agustsson, M. Tschannen, F. Mentzer, R. Timofte, and L. V. Gool, ”Generative adversarial networks for extreme learned image compression,” *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 221-231. [Online]. Available: http://arxiv.org/abs/1804.02958.

[5] S. D. Rane and G. Sapiro, ”Evaluation of JPEG-LS, the new lossless and controlled-lossy still image compression standard, for compression of high-resolution elevation data,” in *IEEE Transactions on Geoscience and Remote Sensing*, vol. 39, no. 10, pp. 2298-2306, Oct. 2001, doi: 10.1109/36.957293.

[6] A. Prabhu, S. Chowdhary, S. J. Narayanan, and B. Perumal, ”Image compression and reconstruction using encoder–decoder convolutional neural network,” in *Intelligent System Design. Springer*, vol. 1171, pp. 845–855, 2021, doi: 10.1007/978-981-15-5400-1-80.

[7] C.-H. Yeh, C.-H. Lin, M.-H. Lin, L.-W. Kang, C.-H. Huang, and M.-J. Chen, ”Deep learning-based compressed

image artifacts reduction based on multi-scale image fusion," in *Information Fusion*, vol. 67, pp. 195–207, March 2021, doi: 10.1016/j.inffus.2020.10.016.

[8] J. Liu, G. Lu, Z. Hu, and D. Xu, "A unified end-to-end framework for efficient deep image compression," *arXiv preprint arXiv:2002.03370*, 2020.

[9] A. G. Ororbia, *et al.,* "Learned Neural Iterative Decoding for Lossy Image Compression Systems," 2019 Data Compression Conference (DCC)," in *2019 Data Compression Conference (DCC)*, 2019, pp. 3-12, doi: 10.1109/DCC.2019.00008.

[10] K. Geetha, V. Anitha, M. Elhoseny, S. Kathiresan, P. Shamsolmoali, and M. M. Selim, "An evolution-ary lion optimization algorithm-based image compression technique for biomedical applications," *Expert Systems*, vol. 38, no. 1, p. e12508, 2021, doi: 10.1111/exsy.12508.

[11] J. Lorandel, H. Lahdhiri, E. Bourdel, S. Monteleone, and M. Palesi, "Efficient compression technique for noc-based deep neural network accelerators," *2020 23rd Euromicro Conference on Digital System Design (DSD)*, 2020, pp. 174-179, doi: 10.1109/DSD51259.2020.00037.

[12] C. Raghavendra, S. Sivasubramanian, and A. Kumaravel, "Improved image compression using effective lossless compression technique," *Cluster Computing*, vol. 22, no. 2, pp. 3911–3916, 2019, doi: 10.1007/s10586-018-2508-1.

[13] Y. Zhang and D. A. Adjeroh, "Prediction by Partial Approximate Matching for Lossless Image Compression," in *IEEE Transactions on Image Processing*, vol. 17, no. 6, pp. 924-935, June 2008, doi: 10.1109/TIP.2008.920772.

[14] N. Akash Bharadwaj, C. S. Rao, Rahul and C. Gururaj, "Optimized Data Compression through Effective Analysis of JPEG Standard," in *2021 International Conference on Emerging Smart Computing and Informatics (ESCI)*, 2021, pp. 110-115, doi: 10.1109/ESCI50559.2021.9396904.

[15] I. Jumakulyyev and T. Schultz, "Lossless PDE-based Compression of 3D Medical Images," in *SSVM*, 2021. pp. 450-462.

[16] Z. Wang, A. C. Bovik, H. R. Sheikh and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," in *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600-612, April 2004, doi: 10.1109/TIP.2003.819861.

[17] P. V. S. Rajan and A. L. Fred, "An efficient compound image compression using optimal discrete wavelet transform and run length encoding techniques," *Journal of Intelligent Systems*, vol. 28, no. 1, pp. 87–101, 2019, doi: 10.1515/jisys-2016-0096.

[18] K. M. Hashim, S. S. Baawi, and B. K. Hilal, "A new proposed method for a statistical rules-based digital image compression," in *Journal of Physics: Conference Series, IOP Publishing*, vol. 1897, no. 1, p. 012067, 2021.

[19] S. Boopathiraja and P. Kalavathi, "A near lossless three-dimensional medical image compression technique using 3d-discrete wavelet transform," in *International Journal of Biomedical Engineering and Technology*, vol. 35, no. 3, pp. 191–206, March 2021, doi: 10.1504/IJBET.2021.113731.

[20] I. M. Pu, "*Fundamental Data Compression*," Oxford: Butterworth-Heinemann, 2005. [Online]. Available: http://www.sciencedirect.com/science/article/pii/B9780750663106500040.

[21] A. K. Jain, "Image data compression: A review," in *Proceedings of the IEEE*, vol. 69, no. 3, pp. 349-389, March 1981, doi: 10.1109/PROC.1981.11971.

[22] B. Subramanian, K. Palanisamy, and V. S. Prasath, "On a hybrid lossless compression technique for three- dimensional medical images," *Journal of applied clinical medical physics*, May 2021, doi: 10.1002/acm2.12960.

[23] M. Yang and N. Bourbakis, "An overview of lossless digital image compression techniques," in *48th Midwest Symposium on Circuits and Systems, 2005*, 2005, pp. 1099-1102 Vol. 2, doi: 10.1109/MWSCAS.2005.1594297.

[24] A. B. Watson, "Image compression using the discrete cosine transform," in *Mathematica Journal*, vol. 4, no. 1, pp. 81–88, 1994.

[25] G. Lakhani, "Optimal Huffman coding of DCT blocks," in *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 14, no. 4, pp. 522-527, April 2004, doi: 10.1109/TCSVT.2004.825565.

[26] A. J. Qasim, R. Din, and F. Q. A. Alyousuf, "Review on techniques and file formats of image compres- sion," *Bulletin of Electrical Engineering and Informatics*, vol. 9, no. 2, pp. 602–610, April 2020, doi: 10.11591/eei.v9i2.2085.

[27] X. O. Zhao and Z. H. He, "Lossless image compression using super-spatial structure prediction," in *IEEE Signal Processing Letters*, vol. 17, no. 4, pp. 383-386, April 2010, doi: 10.1109/LSP.2010.2040925.

[28] M. I. Khalil, "Image compression using new entropy coder,," in *International Journal of Computer Theory and Engineering*, vol. 2, no. 1, pp. 39–41, February 2010.

[29] M. Hasan and K. Nur, "A lossless image compression technique using location based approach," in *Inter- national Journal of Scientific Technology Research*, vol. 1, no. 2, 101–105, March 2012.

[30] L. D. Demaret and A. Iske, "Advances in digital image compression by adaptive thinning," *Annals of the MCFA*, vol. 3, pp. 105–109, 2004.

[31] N. Azman, S. Ali, R. A. Rashid, F. A. Saparudin, and M. A. Sarijari, "A hybrid predictive technique for lossless image compression," *Bulletin of Electrical Engineering and Informatics*, vol. 8, no. 4, pp. 1289–1296, December 2019, doi: 10.11591/eei.v8i4.1612.

[32] K. Karadimitriou and J. M. Tyler, "Min-max compression methods for medical image databases," *ACM SIGMOD*

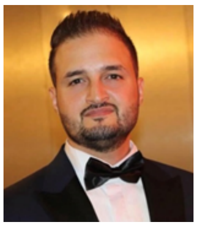
Record, vol. 26, no. 1, pp. 44-52, March 1997, doi: 10.1145/248603.248613.

[33] E. Yang and L. Wang, ”Joint optimization of run-length coding, huffman coding, and quantization table with complete baseline jpeg decoder compatibility,” in *IEEE Transactions on Image Processing*, vol. 18, no. 1, pp. 63-74, Jan. 2009, doi: 10.1109/TIP.2008.2007609.

[34] A. Avramović, ”Lossless compression of medical images based on gradient edge detection” *2011 19th Telecommunications Forum (TELFOR) Proceedings of Papers*, 2011, pp. 1199-1202, doi: 10.1109/TELFOR.2011.6143765.

[35] H. Imane, B. Mohammed, and B. Ahmed, ”Hybrid medical image compression method using quincunx wavelet and geometric actif contour,” *Bulletin of Electrical Engineering and Informatics*, vol. 9, no. 1, pp. 146–159, February 2020, doi: 10.11591/eei.v9i1.1675.

BIOGRAPHIES OF AUTHORS



Abdel Rahman Idrais (Student Member, IEEE) received the B.Sc. and M.Sc. degrees in Computer Engineering from Jordan University of Science and Technology, Irbid, Jordan, in 2011 and 2015, respectively. He is currently working toward the Ph.D. degree in Electrical and Computer Engineering with Concordia University, Montreal, QC, Canada, under supervision of Dr. M. Omair Ahmad and Dr. M.N.S. Swamy. His research interests include machine and deep learning, image processing, computer vision.

Inad Aljarrah is an Associate Professor at the Department of Computer Engineering at Jordan University of Science and Technology. He received his B.S. in Electrical Engineering from Jordan University of Science and Technology in 1999. He received his Masters and Ph.D. degrees in Electrical and Computer Engineering from Ohio University, Athens, Ohio, USA in the years 2002 and 2006 respectively. His research interests are Computer Vision, Image Processing and Analysis, and Artificial Intelligent systems. Currently he is a visiting scholar at the Electrical and Computer engineering department at North Carolina State University Raleigh, NC.

Osama Al-Khaleel is an associate professor of Computer Engineering in the Department of Computer Engineering at Jordan University of Science and Technology (Irbid, Jordan), received his B.S in Electrical Engineering from Jordan University of Science and Technology in 1999, M.Sc. and Ph.D. in Computer Engineering from Case Western Reserve University, Cleveland, OH, USA in 2003 and 2006, respectively. Currently, his main research interests are in embedded systems design, reconfigurable computing, computer arithmetic, and logic design.