PIXEL SIZE REDUCTION LOSS-LESS IMAGE COMPRESSION ALGORITHM

Pralhadrao V Shantagiri[1] and Saravanan K N[2]

[1]MPhil Scholar, Department of Computer Science, Christ University, Bangalore, India
gpralhadrao@gmail.com
[2]Asst. Professor, Department of Computer Science, Christ University, Bangalore, India
saravanan.kn@christuniversity.in

ABSTRACT

The digital signal processing and multimedia Computing is used to produce and process a large number images. Storing of raw image takes more space on storage device and more bandwidth over network during transmission. Very few image compression algorithms exist in loss-less image compression category to achieve compression ratio without transforming the image from the spatial domain. This paper proposes new spatial domain loss-less image compression algorithm for synthetic color images of 24 bits. The proposed algorithm does compression of an image by reducing the size of pixel. The size of pixel is reduced by representing pixel using only required number of bits instead of 8 bits per color. The pre-processing step takes care of re-valuing pixel based on occurrence to get better compression ratio. The proposed algorithm has been applied on set of test images and the results obtained are encouraging. The compression ratio has been compared with Huffman, TIFF, PPM-Tree, GPPM, PCX [19] and found better.

KEYWORDS

Pixel Size Reduction, Bits Per Pixel, Run Length Encoding, Huffman Encoder, Loss-less Image Compression & Compression Ratio

1. INTRODUCTION

The process of reducing the amount of data needed for storage and transmission of given image on storage place is image compression. The compression helps to reduce transmission bandwidth in case of network or satellite transmission [1]. The proposed Pixel Size Reduction loss-less image compression algorithm will retain the originality of an image when it is decompressed. We have chosen synthetic images to validate the proposed algorithm but can be used to test images of other types like photo, wallpaper, medical images, satellite images and etc.

The image is represented as a matrix of pixel value and each pixel for each color is represented in integer value ranging from 0 to 255 occupying 8bits. The RGB image has 3 planes representing three primary colours namely Red, Green and Blue color as components of an image. The test images used to test the proposed algorithm are 24 bits colour.

2. PAGE LAYOUT

This paper is organized as follows. The next section, Section 3, describes the different image compression methods and calculating CF of compressed image. In Section 4, we described the proposed PSR algorithm steps for both encoding and decoding of an image.

In Section 5 shows the experimental results obtained by using proposed PSR algorithm and its analysis. In Section 6 includes conclusion of research work carried out.

3. IMAGE COMPRESSION

The technological advancement in the field of information technology has led the development in many fields. This includes Internet, Teleconferencing, Medical, Military, Multimedia, High Definition Television Technologies, space, whether forecasting and etc. Because of this, the quantity of image data being handled by computers has increased rapidly. This made need of image compression, to save storage space, to reduce network bandwidth [1] consumption during transmission of an image over network. The image compression helps to save space on storage disks and also usage of Internet for transmission. The bigger image takes more time to transmit. Hence, the need of an image compression.

3.1. REDUNDANCY IN AN IMAGE

Every image will have redundant data. The redundancy means duplication of information across the image. Either it may be repeating pixel across the image or pattern, which is repeating more frequently in the image. The image compression happens by taking advantage of this redundant information [2,3] in the image. The redundancy is quality of every image irrespective of which category that image belongs. Reduction of redundancy helps to achieve saving of storage space of an image.

Neighbouring pixels are not statistically independent [3, 4]. It is due to the correlation between the neighbouring pixels of an image. This is called Inter-pixel redundancy. If neighbouring pixels are statistically correlated and they are not required to represent independently in am image, then it becomes spatial redundancy in am image. In spatial redundancy, pixels are predicted from their neighbours.

Another kind of redundancy in an image is coding redundancy, which is associated with the representation of information in the form codes like Huffman, arithmetic and rice codes. These codes must be efficient in order to compress the image effectively.

3.2. TYPES OF IMAGE COMPRESSION

The lossy and loss-less are types of image compression techniques. In general, lossy techniques provide for greater compression ratios than loss-less techniques. The proposed algorithm belongs loss-less technique. The loss-less compression can be one of the following:

- Predictive
- Transform based coding
- Dictionary based
- Ad hoc

The proposed algorithm falls under ‘ad-hoc’. The following are some of the loss-less compression coding.

3.2.1. RUN LENGTH ENCODING

The basic algorithm in loss-less image compression used for sequential data. It is more suitable for binary images. Usually consecutive pixels in a smooth region of an image are identical or the small variation among them. It replaces sequences of pixels of same value with occurrence and pixel [3]. The Run Length Encoding algorithm is briefly shown in Figure 1.

142 142 142 142 230 230 230 230 230 230 121 121 121 121
{142, 4} {230, 6} {121, 4}

Figure 1. Depicting Run-Length Encoding

3.2.2. STATIC HUFFMAN ENCODING

The idea behind Huffman static coding [4, 5] is, representing frequently occurring pixel in shorter codeword instead of representing fixed length codeword based on frequency of each pixel. The code words are stored in Huffman-Tree. Based on this tree, pixels are presented with shortest code word as applicable.

3.2.3. ADAPTIVE HUFFMAN ENCODING

This is dynamic version of Huffman encoding technique and avoids preparation of frequency table to know the occurrence of each pixel in an image at the beginning [6, 7]. It calculates as it reads after each pixel. The frequency of each symbol is called weight of that symbol and this is done at stage.

3.2.4. LEMPEL–ZIV–WELCH (LZW)

This is a dictionary based encoding. This algorithm basically replaces strings of pixels with single codes and creates dynamic dictionary of codes without doing any analysis of the incoming data [8, 22]. It keeps on updating table by adding every new string of pixels it reads. It outputs singe code instead of string of pixels. The default size of table is 256 to hold pixel values from 0 to 255 for 8 bits. The table can be extended up to 4095 based on new pattern strings it forms in the process. This can take up to 12 bits.

3.2.5. ARITHMETIC ENCODING

In this technique, instead of coding each symbol separately, whole image sequence is coded with a single code [9]. The longer the message, the smaller the interval to represent the message becomes. More probable symbols reduce the interval less than the less probable symbols and hence add fewer bits in the encoded message. As a result, the coding result can reach to Shannon's entropy limit for a sufficiently large sequence of input symbols as long as the statistics are accurate. Arithmetic coding is based on the following principle [10]. Given that

- The symbol alphabet is finite;
- All possible symbols sequences of a given length are finite;
- All possible sequences are count-ably infinite;
- The number of real numbers in the interval (0, 1) is unaccountably infinite; we can assign a unique subinterval for any given input (sequence of symbols).

3.2.6. THE PEANO COUNT TREE (P-TREE)

This is a type of ad-hoc coding which is based on quad-tree concept [11]. The P-trees are a loss-less, compressed, and data-mini-greedy data structure falling under spatial domain compression. It recursively divide the entire spatial data into quadrants and record the count of 1-bits for each quadrant, thus forming a quadrant count tree [12]. Using P-tree structure, all the count information can be calculated quickly.

Pure-0 quadrant: Composed of only ZERO.
Pure-1 quadrant: Composed of only ONE.

The P-Tree loss-less compression helps to achieve better compression when more number of consecutive 1's and 0's are present in the image in multiple of four.

3.2.7. THE PATTERN MASK TREE (PM-TREE)

It is an enhanced version of P-Tree, which masks, rather than counts, are used. In a PM-tree, we use three-value logic to represent pure-1, pure-0, and mixed quadrants [14]. This also belongs to quad-tree family. The mixed quadrant allows having better compression rate compared to P-Tree.

3.2.8. THE PEANO PATTERN MASK TREE (PPM-TREE)

It is an enhanced version of PM-Tree, which uses four-value logic and the fourth logic to represent pure-1, pure-0, mixed quadrants and **P** for pattern [14]. This pattern is dynamic values depending on the tree constructed. In fact, the pattern state is a special case of a mixed state. The dynamic pattern **P** helps to achieve good compression rate compared to PM-Tree.

3.2.9. GENERIC PEANO PATTERN MASK TREE (GPPM TREE)

The Generic PPM tree is an enhanced version of PPM-Tree, which overcomes the draw back of P-Tree and PMM-Tree for low compression because of lower order bits are compressed the neighbouring bits and may not share the same values with high probability. When probability of sharing the same bit value is low, Pure-0 and Pure-1 are impossible. Hence to over-come this, GPPM-Tree introduced 3 dynamic patterns in place of Pure-1, Pure-0 and P [15, 16]. The dynamic patterns are P1, P2 and P3.

3.3. MEASURING COMPRESSED IMAGE

The compression factor is used to numerically quantify the amount of data reduction in numbers. The compression factor is calculated based on uncompressed original size of an image and size of an image after compression [3]. Following is the formula to compute CF:

Compression factor (CF) = uncompressed file size / compressed file size
Another way of measuring algorithm's performance is to find average number of bits required to encode single symbol, which is called as compression rate or Bits Per Pixel (BPP).

BPP = 8 / CF.

Each algorithm works best on certain kind of image. Algorithm cannot be applied on all kind of images, which results into disaster. Hence, it is very important to identify the kind of images it is suitable and gives best result compared to other algorithm.

4. PROPOSED PIXEL SIZE REDUCTION LOSS-LESS ALGORITHM

The encoding part of proposed Pixel Size Reduction image compression algorithm includes 3 main tasks. The first part performs preparation of pixel occurrence table for each colour component of 24-bits RGB image and stores in order. The second task is to do, re-valuing the pixel based maximum occurrence to 0, next occurrence to 1 and so on and representing pixel with least binary bits instead of 8 bits per pixel. While revaluing pixel, prepare header information for each pixel to help reconstruct pixel during decoding. Last step is to compress header pixels formed by Pixel Size Reduction, using LZW encoder and write out the compressed file.

4.1. COMPRESSION ALGORITHM

The proposed Pixel Size Reduction loss-less image-compression algorithm works by simply representing pixel in least number of binary bits from each symbol. It makes sure that; maximum occurring symbol will become 0, next maximum occurring symbol will become 1 and so on till 255.

Algorithm:

Step 01: Read image if empty go to Step 10
Step 02: Prepare Pixel Occurrence table and have in order
Step 03: Repeat Steps 4 thru 7 till EOF.
Step 04: Read each Pixel
Step 05: Get its re-valued pixel
Step 06: Put Pixel in minimum bits. Like Decimal 9 should be in binary 1001 instead of 00001001
Step 07: Find out length of minimum bits and store it as header information for pixel
Step 08: Compress 'header information' using LZW algorithm
Step 09: Write out pixels bits of least size and compressed header into file.
Step 10: stop.

4.2. DECOMPRESSION ALGORITHM

It is the process of un-compressing the compressed image using proposed Pixel Size Reduction loss-less image compression algorithm. The decoding process starts by reading compressed image, and constructing Pixel Occurrence table, decompressing header using LZW decoder and decompression actual data using header information. The algorithm to decode the compressed file is depicted in the next section.

Algorithm:

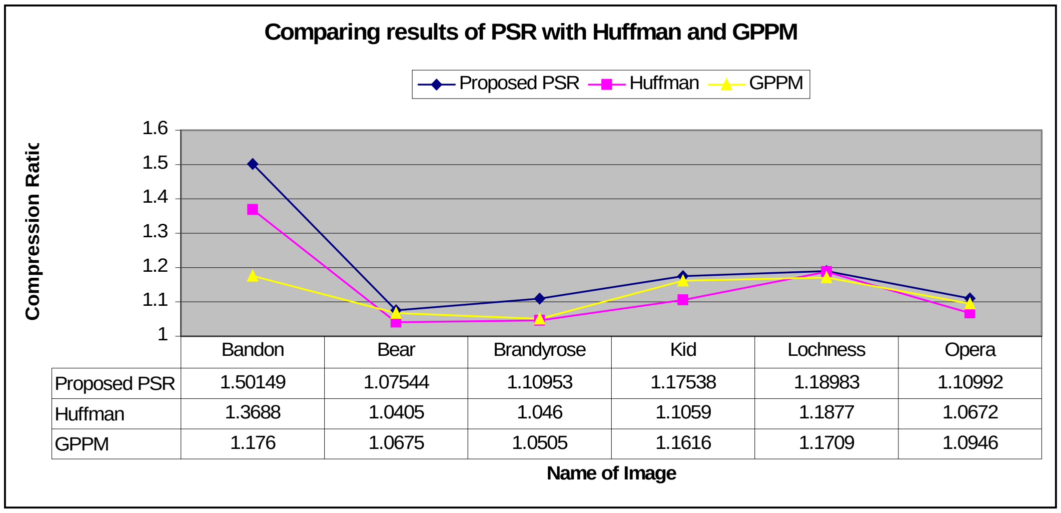
Step 01: Read compressed image file
Step 02: If file is empty go to step 9 otherwise Step 3
Step 03: Prepare pixel re-valuing table
Step 04: Decode header info using lzw decoder algorithm
Step 05: Repeat Steps 6 thru 8 till EOF otherwise go to step 9
Step 06: Read symbols from encoded file
Step 07: Read length of pixel from header info and re-construct pixel

Step 08: Get original pixel from Pixel Re-valuing table
Step 09: Write into decoded file.
Step 10: Stop.

5. Experimental Results

The performance of proposed PSR encoding and decoding algorithm experimented on few hundreds of BMP images selected for experiment purpose did better. The 2 sets of image results have been shown in this paper. The image sets include 24-bits RGB images. The proposed PSR algorithm performs better compare to other algorithms listed in Figure 2 and Figure 3.

Figure 2, the compression ratio result of Huffman and GPPM algorithms have been taken from [15]. The Huffman and GPPM also are loss-less compression techniques. These results are being compared with proposed PSR algorithm. The experimental results of Figure 2, show that, proposed PSR algorithm performs better than Huffman and GPPM. The compression ratio of PSR, Huffman and GPPM has been shown along with line graph in Figure 2.

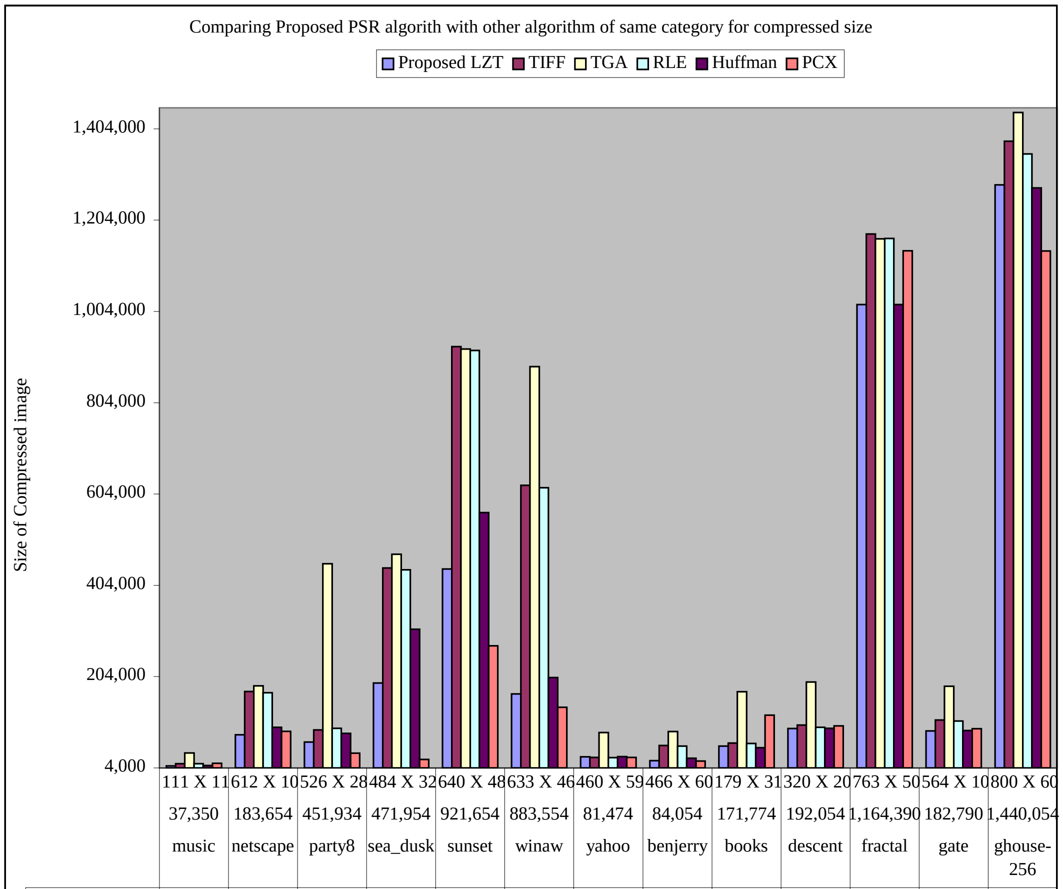


	Bandon	Bear	Brandyrose	Kid	Lochness	Opera
Proposed PSR	1.50149	1.07544	1.10953	1.17538	1.18983	1.10992
Huffman	1.3688	1.0405	1.046	1.1059	1.1877	1.0672
GPPM	1.176	1.0675	1.0505	1.1616	1.1709	1.0946

.

Figure 2. Comparing proposed PSR algorithm with Huffman [15] and GPPM [15] for images of size 512 X 512

The Figure 3, includes images of type synthetic. It includes name of image, size of an image, compressed size, original size of an image. The images chosen are of different dimension. In Figure 3, the proposed PSR has been compared with TIFF, TGA, RLE, Huffman and PCX loss-less compression algorithms. The TIFF results are generated using Windows XP's 'paint' program and other algorithm results are generated using Image Magic tool from Linux platform.
The proposed PSR algorithm performed better in terms of compressed size compared to TIFF, TGA, RLE, Huffman and PCX in most of the cases as shown in Figure 3. In few images, PSR performed close to PCX especially in sea_dusk, sunset and, yahoo images.



	111 X 11	612 X 10	526 X 28	484 X 32	640 X 48	633 X 46	460 X 59	466 X 60	179 X 31	320 X 20	763 X 50	564 X 10	800 X 60
	37,350	183,654	451,934	471,954	921,654	883,554	81,474	84,054	171,774	192,054	1,164,390	182,790	1,440,054
	music	netscape	party8	sea_dusk	sunset	winaw	yahoo	benjerry	books	descent	fractal	gate	ghouse-256
🟦 Proposed LZT	8,363	76,786	60,779	190,121	439,912	165,961	28,140	19,871	51,615	90,426	1,019,24	84,953	1,281,42
🟥 TIFF	13,318	171,414	87,298	442,020	926,798	623,026	27,120	53,120	58,602	97,736	1,173,47	108,870	1,376,89
🟨 TGA	36,981	183,618	451,326	471,918	921,618	883,053	81,438	83,898	170,784	192,018	1,162,83	182,754	1,440,01
🟦 RLE	13,331	168,747	90,848	437,846	918,406	617,669	26,665	51,868	57,461	93,040	1,163,82	106,958	1,349,15
🟪 Huffman	9,442	93,154	79,612	307,741	563,286	202,159	28,848	25,416	48,197	90,627	1,019,29	85,792	1,274,80
🟥 PCX	14,403	84,212	36,519	22,383	271,622	136,735	27,116	19,074	119,794	96,267	1,136,93	89,753	1,136,49

Figure 3. Comparison of Synthetic [21] compressed image of proposed PSR with TIFF, TGA, RLE, Huffman and PCX.

6. CONCLUSIONS

In this paper, we introduced the principles of Pixel Size Reduction (PSR) image compression loss-less image compression algorithm. We also had shown the procedures of compression and decompression of proposed PSR loss-less image compression algorithm. The PSR performed better in compressing 24-bits RGB images including Synthetic color images compare to other algorithms shown in the results. This algorithm has been experimented on thousands of images and shown sample result in this paper. It performed well in synthetic image set in almost all cases and come close to PCX in few cases. The proposed algorithm performs better when highest occurring unique pixels are more.

As for the direction of future studies, it might be interesting topics to consider using this method in other Tree based loss-less image compression algorithms.

REFERENCES

[1] Sachin Dhawan, “A Review of Image Compression and Comparison of its Algorithms”, International Journal of electronics & Communication technology, vol. 2, no. 1, pp. 22-26, March 2011.

[2] Sindhu M and Rajkamal R, “Images and Its Compression Techniques – A Review”, ACEEE, International Journal of Recent Trends in Engineering, vol 2, no. 4, pp. 71-75, November 2009.

[3] S Jayaraman, S Esakkirajan and T Veerakumar, Digital Image Processing, 3rd Reprint, Tata McGraw Hill, 2010.

[4] Ida Mengyi Pu, Fundamental Data Compression, 1st ed., Butterworth-Heinemann publications, 2006.

[5] Jagadish H Pujar, “A New Lossless Method of Image Compression And Decompression Using Huffman Coding Techniques”, Journal of Theoretical and Applied Information Technology, vol. 15, no. 1, pp. 18-23, 2010.

[6] Jeeffrey Scott Vitter, "Dynamic Huffman Coding", ACM Transactions on Mathematical Software, vol. 15, no. 2, pp. 158-167, June 1989.

[7] Jeffrey Scott Vitter, "Design and Analysis of Dynamic Huffman Codes", Journal of the Association for Computing Machinery, vol. 34, no. 4, pp. 825-845, October 1987.

[8] Nishad PM, R.Manicka Chezian, "Enhanced LZW (Lempel-Ziv-Welch) Algorithm by Binary Search with Multiple Dictionary to Reduce Time Complexity for Dictionary Creation in Encoding and Decoding", International Journal of Advanced Research in Computer Science and Software Engineering, vol. 2, no. 3, pp. 192-198, March 2012.

[9] Ian H Willen, Radford M Neal, and John G Cleary, "Arithmetic Coding For Data Compression", Communications of the ACM, vol. 30, no. 6, pp. 520-540, June 1987.

[10] Khalid Sayood, Lossless Compression Handbook, ACADEMIC PRESS, 2003.

[11] Fazle Rabbi, Mohammad Hossain, et. al. “Lossless Image Compression using P-tree”, in Proceedings of ICCIT 2003, Dhaka, Bangladesh.

[12] Li-yangchun, “Image Compression Based on P-tree”, in 2nd International Conference Information Science and Engineering (ICISE), pp. 4550-4553, Dec. 4-6, 2010.

[13] Xiaobo Li, Jason Knipe, Howard Cheng, “Image compression and encryption using tree structures”, Pattern Recognition Letters, vol. 18, no. 11-13, pp. 1253-1259, November, 1997.

[14] Shams Mahmood Imam, S M Rezaul Hoque, et. al, “A New Technique of Lossless Image Compression using PPM-Tree”, in Proceedings of 8th International Conference of Computer and Information Technology (ICCIT), Dhaka, Bangladesh, 2005.

[15] Dr. M.Mohamed Sathik, S.Radhakrishnan and Dr. R K Selvakumar, “GPPM Tree in CFA Image Compression”, International Journal of Electronics & Communication Technology, vol. 2, no. 3, pp. 84-87, September 2011.

[16] Mohammad Kabir Hossain, Shams MImam, Khondker Shajadul Hasan and William Perrizo, “A Lossless Image Compression Technique Using Generic Peano Pattern Mask Tree”, in Proceedings of 11th International Conference on Computer and Information Technology (ICCIT 2008), pp. 317-322, Dec. 24-27, 2008.

[17] Terry A Welch, "A Technique for High Performance Data Compression", IEEE Computer, vol. 17, no. 6, pp. 8-19, June 1984

[18] Ming Yang, Nikolaos Bourbakis, “An Overview of Loss-less Digital Image Compression Techniques”, in 48th Midwest Symposium on Circuits and Systems, pp. 1099-1102, Aug 7-10, 2005.

[19] Che-Chern Lin, “A Modified PCX Image Compression Algorithm”, 7th WSEAS Int. Conf. on Applied Computer & Applied Computational Science (ACACOS '08), Hangzhou, China, pp. 740-745, April 6-8, 2008.

[20] Sindhu M, Rajkamal R, “Images and Its Compression Techniques –A Review”, International Journal of Recent Trends in Engineering, vol. 2, no. 4, pp. 71-75, November 2009.

[21] “The Synthetic image database”, ftp://ftp.ieeta.pt/~ap/images/synthetic/, Feb 25, 2012.

[22] Simrandeep kaur, V Sulochana Verma, “Design and Implementation of LZW Data Compression Algorithm”, International Journal of Information Sciences and Techniques, vol. 2, no. 4, pp. 71-81, July 2012.

AUTHORS

Pralhadrao V Shantagiri received the MIT in Information Technology from Manipal University, Karnataka, India and MS in VLSI CAD from Manipal University, Manipal, Karnataka. Presently pursuing M.Phil in Computer Science from Christ University, Karnataka, India and awaiting result of Viva-Voce of his thesis work on Image Compression Technique. Currently, he is working with Synopsys India as R & D Engineer, Bangalore. His topic of interest includes Image Compression, Data Structure, Algorithm Development, VLSI Test and EDA. Email is gpralhadrao@gmail.com.

Saravanan K N has obtained his Bachelor Degree in Science from Madhurai Kamaraj University, Tamil Nadu, India, in 1994, Master Degree in Computer Science from Madhurai Kamaraj University, Tamil Nadu, India, in 1996, Master of Philosophy in Computer Science from Bharathidasan University, Tamilnadu, India in 2006. He is working as a Assistant Professor at Department of Computer Science, Christ University, Karnataka, India. His research interest includes Image Processing, Image Compression and, Data Structure and Algorithm. He enjoys Teaching various subjects in Computer Science. He attended around 11 National and International Conferences and has publication in his name. Email-id is saravanan.kn@christuniversity.in.