International Journal of Applied Engineering Research
ISSN 0973-4562 Volume 10, Number 2 (2015) pp. 4079-4088

http://www.ripublication.com

An Efficient Lossless Image Compression Technique

B.M.S.Rani[1], M.Divya Sree[2],T.Ravi [3]
[1,2]Assistant Professor,Vignan Nirula Institute of technology for women,
[3]Associate Professor,KL University,
ranibms@gmail.com[1] divyasreemikkili@gmail.com[2], raviblind@gmail.com[3]

Abstract

The necessity for an efficient technique for compression of Images increasing because the raw images need large amounts of disk space seems to be a big disadvantage during transmission as well as storage. Even though there are so many compression technique already existing a better technique which is faster, memory efficient and simple surely suits the requirements of the user. In this paper we proposed the Lossless technique of image compression and decompression using a simple coding method called Huffman coding. This technique is simple in implementation and utilizes less memory space. An algorithm has been designed and implemented to compress and decompress the given image using Huffman coding techniques in a MATLAB software.

Keywords: Huffman codes, Huffman coding, memory space, decompression, transmission symbol, lossless.

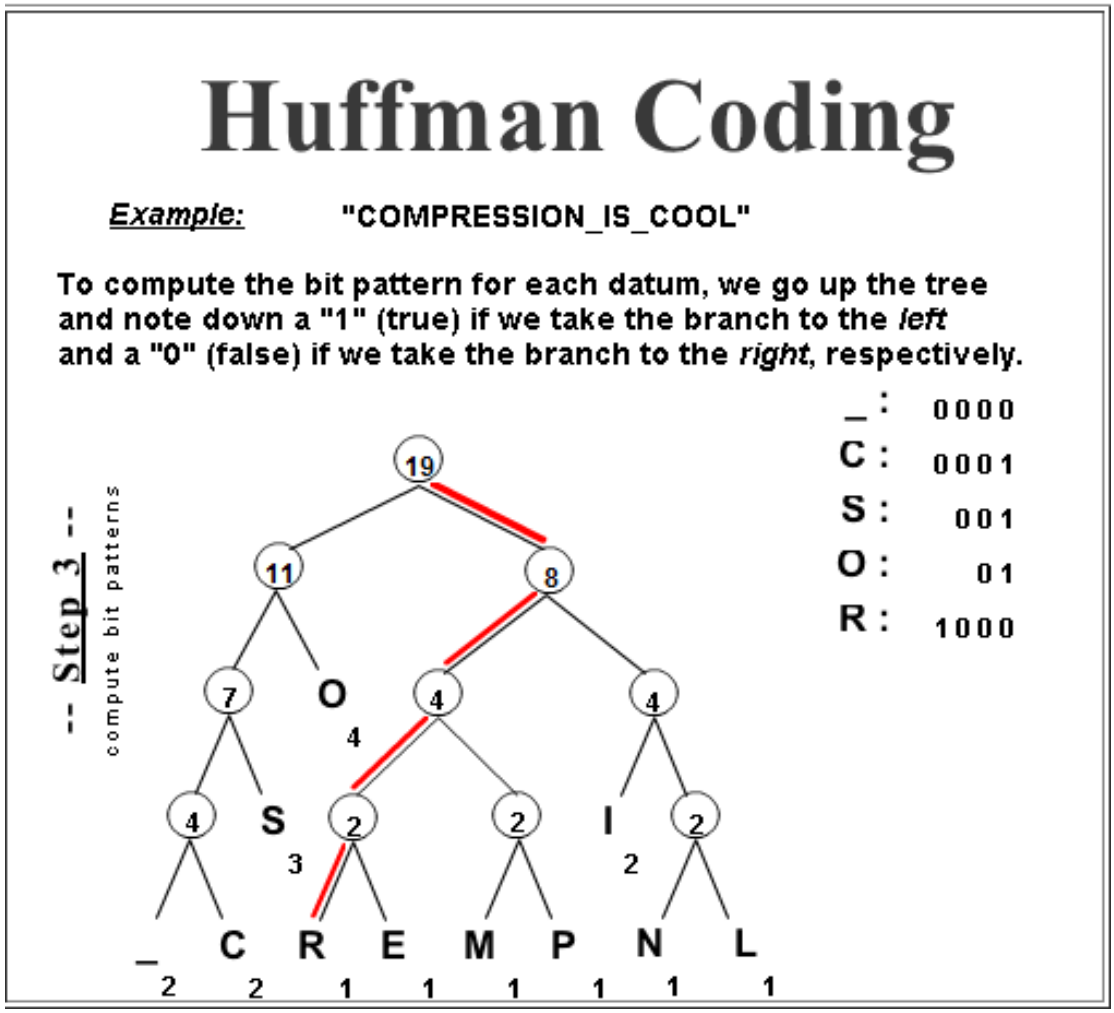
Introduction

Most people have downloaded large files, such as music or video, from the Internet. Because of the large size of these files, downloading them can take hours. To solve this problem, and make better use of disk space, large files are compressed, using various software. Once downloaded, they can then be decompressed, and viewed, using a decompression program Compression is the process of reducing the size of a file by encoding its data information more efficiently. By doing this, the result is a reduction in the number of bits and bytes used to store the information. In effect, a smaller file size is generated in order to achieve a faster transmission of electronic files and a smaller space required for its downloading.

Implementation of Loss Less Compression and Decompression Techniques

Huffman coding

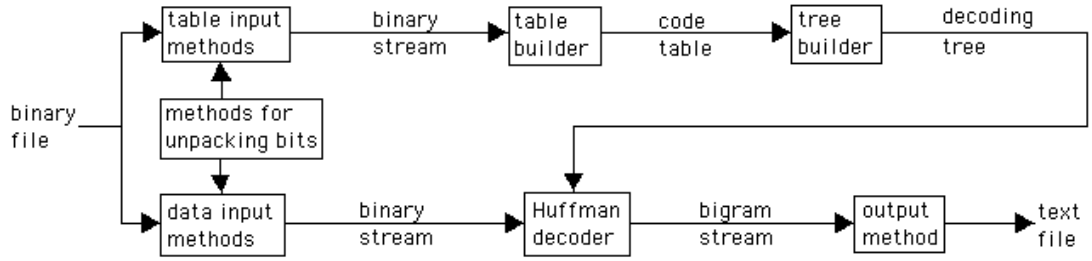
Huffman code procedure is based on the More frequently occurred symbols as they have shorter codes than symbol that occur less frequently and two symbols that occur least frequently will have the same length.The Huffman code is designed by merging the lowest probable symbols and this process is repeated until only two probabilities of two compound symbols are left and thus a code tree is generated and Huffman codes are obtained from labelling of the code tree.



Huffman decoding

After the code has been created, coding and/or decoding is accomplished in a simple look-up table manner. The code itself is an instantaneous uniquely decodable block code. It is called a block code,because each source symbol is mapped into a fixed sequence of code symbols.It is instantaneous, because each code word in a string of

code symbols can be decoded without referencing succeeding symbols. It is uniquely decodable,because any string of code symbols can be decoded in only one way. Thus, any string of Huffman encoded symbols can be decoded by examining the individual symbols of the string in a left to right manner. For the binary code ,a left-to-right scan of the encoded string 010100111100 reveals that the first valid code word is 01010, which is the code for symbol a3. The next valid code is 011, which corresponds to symbol a1. Valid code for the symbol a2 is 1,valid code for the symbols a6 is 00,valid code for the symbol a6 is Continuing in this manner reveals the completely decoded message a5 a2 a6 a4 a3 a1 , so in this manner the original image or data can be decompressed using Huffman decoding as explained above . At first we have as much as the compressor does a probability distribution. The compressor made a code table. The decompressor doesn't use this method though. It instead keeps the whole Huffman binary tree, and of course a pointer to the root to do the recursion process. In our implementation we'll make the tree as usual and then you'll store a pointer to last node in the list, which is the root. Then the process can start. We'll navigate the tree by using the pointers to the children that each node has. This process is done by a recursive function which accepts as a parameter a pointer to the current node, and returns the symbol.



Variations

Many variations of Huffman coding exist, some of which use a Huffman-like algorithm, and others of which find optimal prefix codes (while, for example, putting different restrictions on the output). Note that, in the latter case, the method need not be Huffman-like, and, indeed, need not even be polynomial time. An exhaustive list of papers on Huffman coding and its variations is given by "Code and Parse Trees for Lossless Source Encoding".

N-Ary Huffman coding

The n-ary Huffman algorithm[2] uses the {0, 1, ... , n − 1}alphabet to encode message and build an n-ary tree. This approach was considered by Huffman in his original paper. The same algorithm applies as for binary (n equals 2) codes, except that the n least probable symbols are taken together, instead of just the 2 least probable. Note that for n greater than 2, not all sets of source words can properly form an n-ary tree for Huffman coding. In this case, additional 0-probability place holders must be added. This is because the tree must form an n to 1 contractor; for binary coding, this is a 2 to 1 contractor, and any sized set can form such a contractor. If the number of

source words is congruent to 1 modulo n-1, then the set of source words will form a proper Huffman tree.

Adaptive Huffman coding

A variation called adaptive Huffman coding[1] involves calculating the probabilities dynamically based on recent actual frequencies in the sequence of source symbols, and changing the coding tree structure to match the updated probability estimates. It is used rarely in practice, since the cost of updating the tree makes it slower than optimized adaptive arithmetic coding, that is more flexible and has a better compression.

Huffman template algorithm

Most often, the weights used in implementations of Huffman coding represent numeric probabilities, but the algorithm given above does not require this; it requires only that the weights form a totally ordered commutative monoid, meaning a way to order weig them. The Huffman template algorithm enables one to use any kind of weights (costs, frequencies, pairs of weights, non combining methods (not just addition). Such algorithms can solve other minimization first applied to circuit design.

Length-limited Huffman coding/minimum variance huffman coding:

Length goal is still to achieve a minimum weighted path length, but there is an additional restriction that the length of each codeword must be less than a given constant. The package simple greedy approach very similar to that used by Huffman's algorithm. Its time complexity is O(nL), where L is the maximum length of a codeword. No algorithm is known to solve this problem in linear or linearithmic time, unlike the presorted and unsorted conventional Huffman problems, respectively.

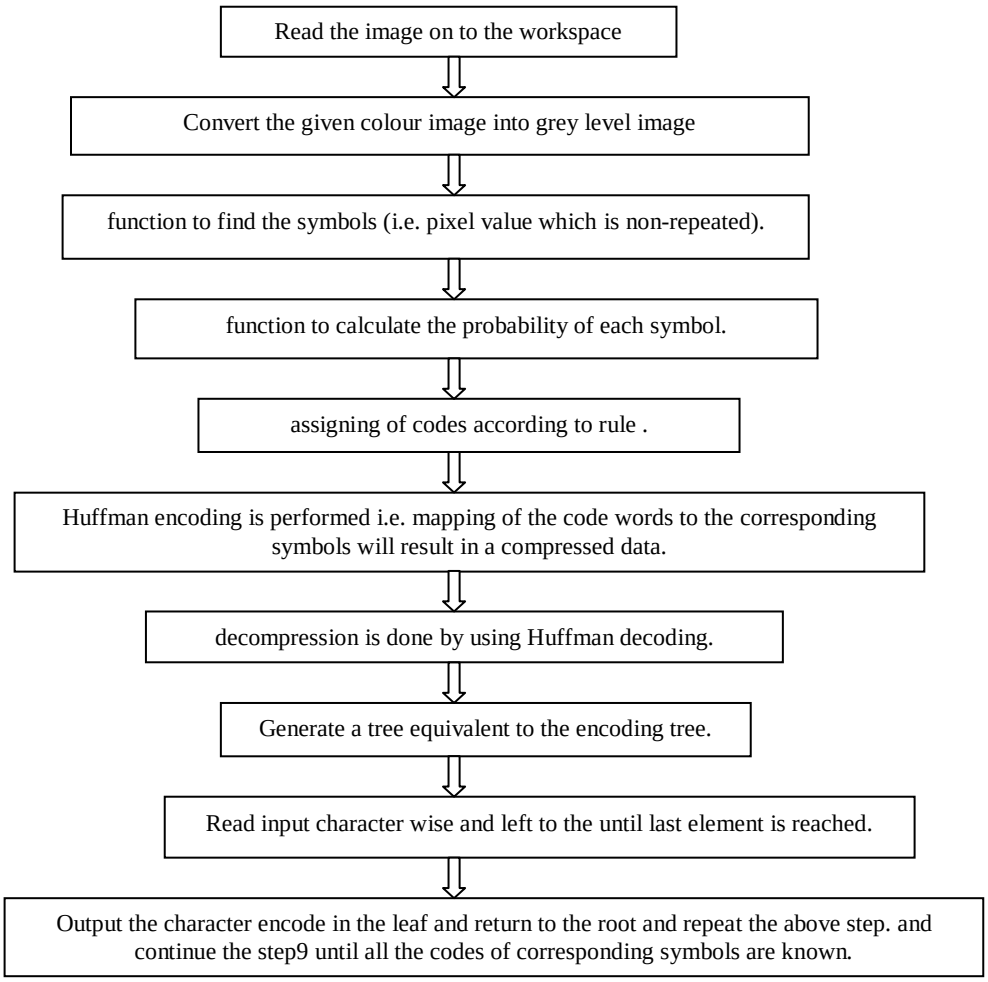
Huffman coding with unequal letter costs:

Assumed that each symbol in the set that the has an equal cost to transmit: a code word whose length is N digits will always have a cost of N, no matter how many of those digits are 0s, how many are 1s, etc. When working under this assumption, minimizing the total cost message and minimizing the total number of digits are the same thing. weights and to add non-numerical weights) and one of many problems, such as minimizing, a problem Length-limited Huffman coding is a variant where the package-merge algorithm solves this problem with a orted In the standard Huffman coding problem, it is code words are constructed from hts roblems, s of the Huffman coding with unequal letter costs is the generalization without this assumption: the letters of the encoding alphabet may have non-uniform lengths, due to characteristics of the transmission medium. An example is the encoding alphabet of Morse code, where a 'dash' takes longer to send than a 'dot', and therefore the cost of a dash in transmission time is higher. The goal is still to minimize the weighted average codeword length, but it is no longer sufficient just to minimize the number of symbols used by the message. No algorithm is known to solve this in the same manner or with the same efficiency as conventional Huffman coding.

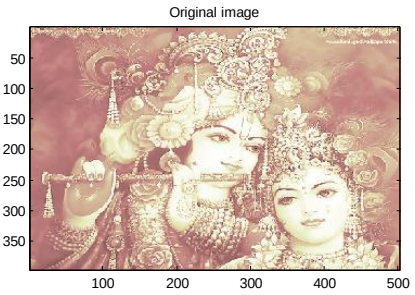
Optimal alphabetic binary trees (Hu–Tucker coding):
In the standard Huffman coding problem, it is assumed that any codeword can correspond to any input symbol. In the alphabetic version, the alphabetic order of inputs and outputs must be identical. Thus, for example, A = {a,b,c} could not be assigned code H(A,C) = {00,1,01}, but instead should be assigned either H(A,C) ={00,01,1} or H(A,C) = {0,10,11}. This is also known as the Hu– Tucker problem, after T. C. Hu and Alan Tucker, the authors of the paper presenting the first linearithmic solution to this optimal binary alphabetic problem, which has some similarities to Huffman algorithm, but is not a variation of this algorithm. These optimal alphabetic binary trees are often used as binary search trees.

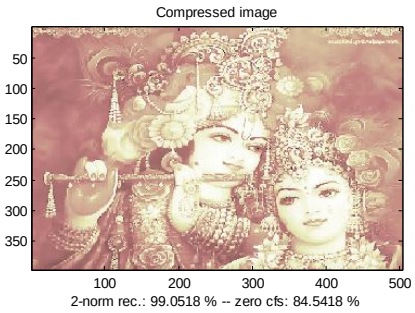
The canonical Huffman code
If weights corresponding to the alphabetically ordered inputs are in numerical order, the Huffman code has the same lengths as the optimal alphabetic code, which can be found from calculating these lengths, rendering Hu–Tucker coding unnecessary. The code resulting from numerically (re-)ordered input is sometimes called the canonical Huffman code and is often the code used in practice, due to ease of encoding/decoding.The technique for finding this code is sometimes called Huffman-Shannon- Fano coding, since it is optimal like Huffman coding, but alphabetic in weight probability, like Shannon-Fano coding. The Huffman-Shannon-Fano code corresponding to the example is \{000,001,01,10,11\}, which, having the same codeword lengths as the original solution, is also optimal. But in canonical Huffman code, the result is \{110,111,00,01,10\}.

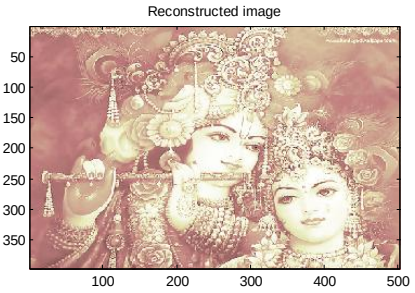
Development of Huffman Coding and Decoding Algorithm

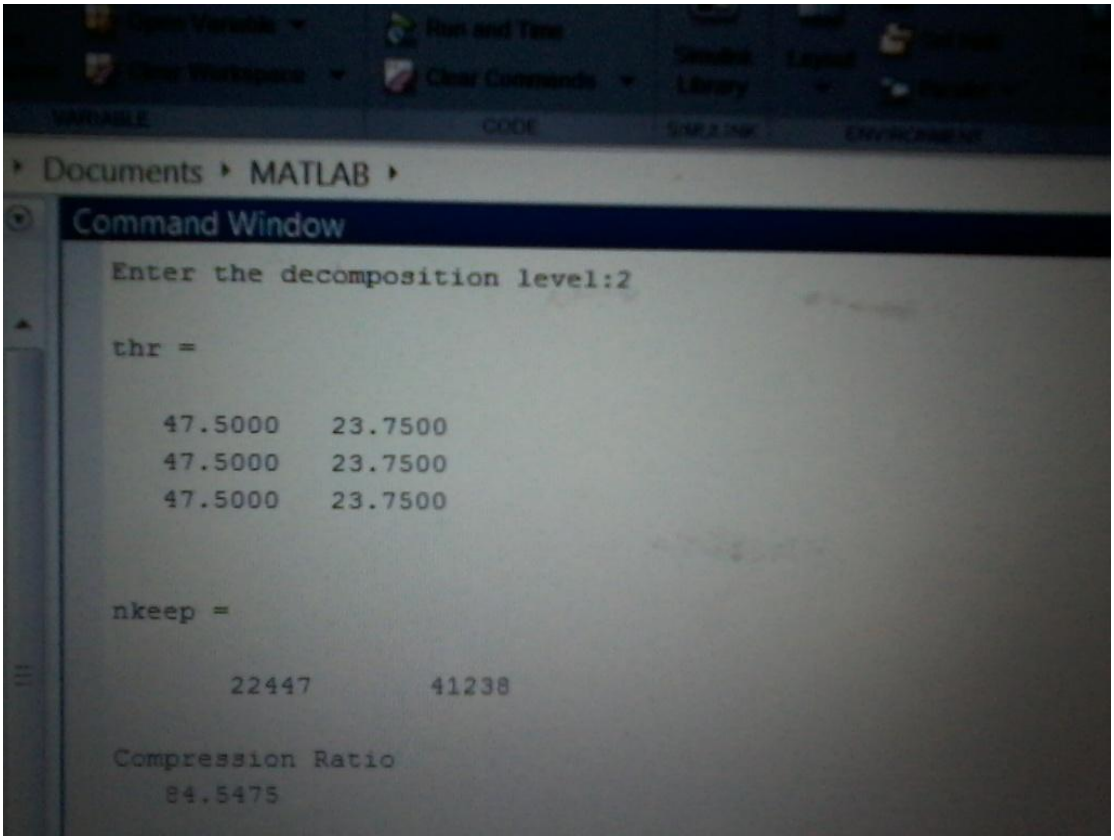


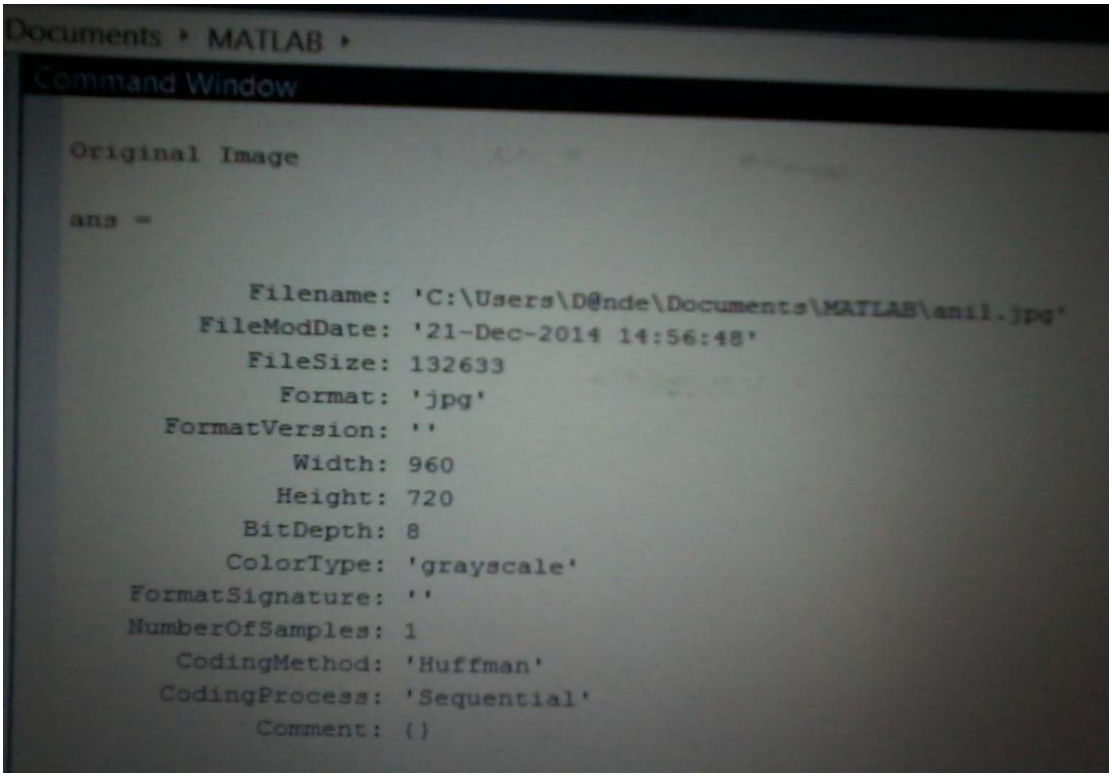
Results

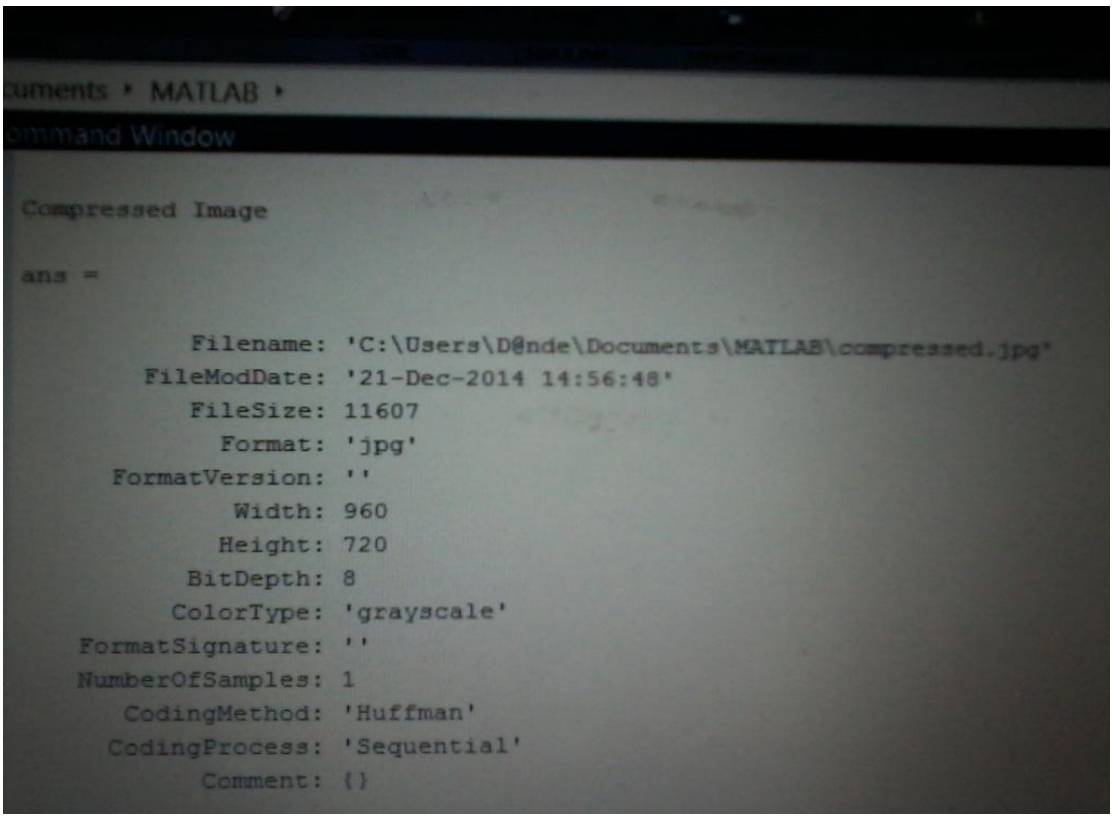

Original image
50
100
150
200
250
300
350
100
200
300
400
500

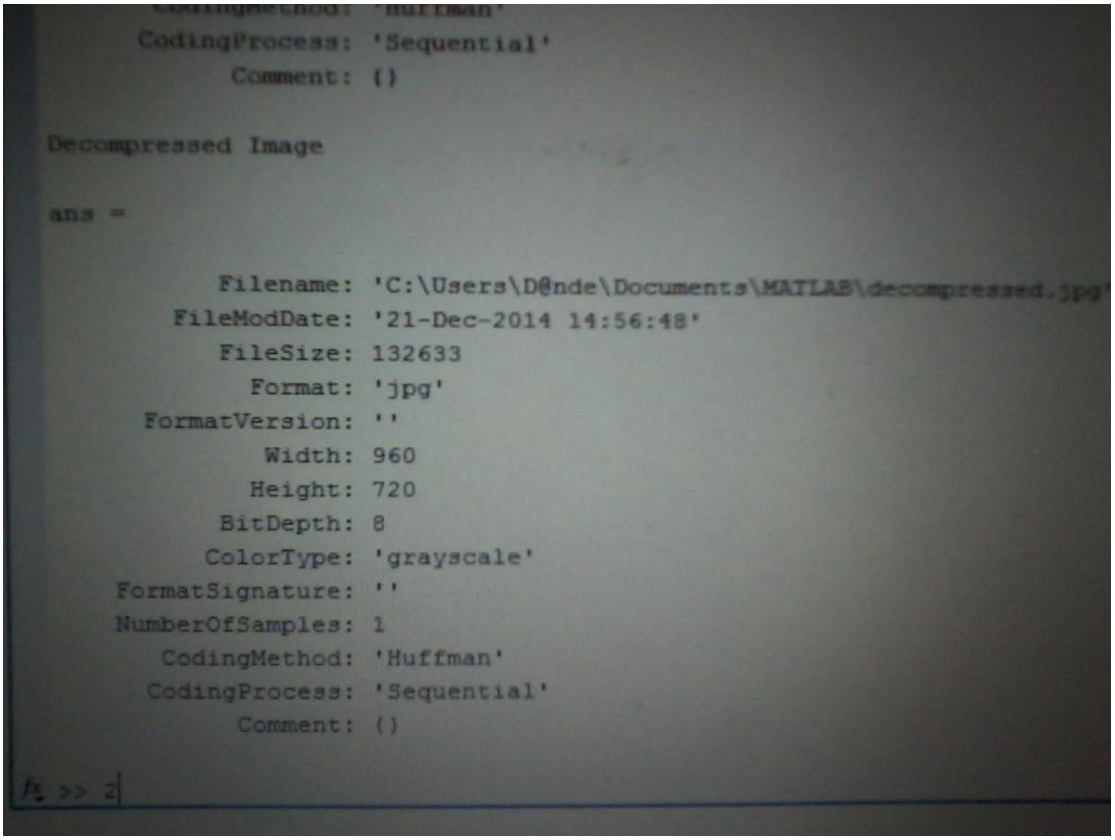

Compressed image
50
100
150
200
250
300
350
100
200
300
400
500
2-norm rec.: 99.0518 % -- zero cfs: 84.5418 %


Reconstructed image
50
100
150
200
250
300
350
100
200
300
400
500


Documents ▸ MATLAB ▸
Command Window
Enter the decomposition level:2
thr =
47.5000 23.7500
47.5000 23.7500
47.5000 23.7500
nkeep =
22447 41238
Compression Ratio
84.5475


Documents ▸ MATLAB ▸
Command Window
Original Image
ans =
Filename: 'C:\Users\D@nde\Documents\MATLAB\anil.jpg'
FileModDate: '21-Dec-2014 14:56:48'
FileSize: 132633
Format: 'jpg'
FormatVersion: ''
Width: 960
Height: 720
BitDepth: 8
ColorType: 'grayscale'
FormatSignature: ''
NumberOfSamples: 1
CodingMethod: 'Huffman'
CodingProcess: 'Sequential'
Comment: {}


uments ▸ MATLAB ▸
ommand Window
Compressed Image
ans =
Filename: 'C:\Users\D@nde\Documents\MATLAB\compressed.jpg'
FileModDate: '21-Dec-2014 14:56:48'
FileSize: 11607
Format: 'jpg'
FormatVersion: ''
Width: 960
Height: 720
BitDepth: 8
ColorType: 'grayscale'
FormatSignature: ''
NumberOfSamples: 1
CodingMethod: 'Huffman'
CodingProcess: 'Sequential'
Comment: {}



The above photos show the compressed and decompression values.

Conclusion

The experiment shows that the higher data redundancy helps to achieve more compression. The above presented a new compression and decompression technique based on Huffman coding and decoding for scan testing to reduce test data volume, test application time. Experimental results show that up to a 0.845compression ratio for the above image is obtained .hence we conclude that Huffman coding is efficient technique for image compression and decompression to some extent. As the future work on compression of images for storing and transmitting images can be done by other lossless methods of image compression because as we have concluded above the result the decompressed image is almost same as that of the input image so that indicates that there is no loss of information during transmission.So other methods of image compression can be carried out as namely JPEG method, Entropy coding, etc.

References

[1] http://en.wikipedia.org/wiki/Huffman_coding

[2] http://en.wikipedia.org/wiki/Adaptive_Huffman_coding
[3] http://en.wikipedia.org/wiki/Block_code
[4] Ternary Tree & FGK Huffman Coding Technique Dr. Pushpa R.Suri † and Madhu Goel Department of Computer Science Applications,Kurukshetra University, Kurukshetra, India
[5] Massachusetts Institute of Technology Department of Electrical Engineering and Computer Science.
[6] Mamta Sharma,‖ Compression Using Huffman Coding‖, IJCSNS International Journal of Computer Science and Network Security, VOL.10 No.5, May 2010,pp 133-141
[7] D.A. Huffman, "A Method for the Construction of Minimum-Redundancy Codes", Proceedings of the I.R.E., September 1952, pp 1098–1102. Huffman's original article.
[8] Talukder, K.H. and Harada, K., A Scheme of Wavelet Based Compression of 2D Image, Proc. IMECS, Hong Kong, pp. 531-536, June 2006.