International Journal of Academic Research
ISSN: 2348-7666 Vol.2, Issue-1(4), January-March, 2015

Efficient JPEG Compression with Hybrid Huffman Coding

Rakoti Chandramohan M.Tech (Final), Department of CSE,Baba Institute of Technology and Sciences (BITS), PM Palem.Madhurawada, Visakhapatnam,A.P,India
V Narayana Rao, Assistant Professor, Departmentof CSE,Baba Institute of Technology and Sciences (BITS), PM Palem.Madhurawada, Visakhapatnam,A.P,India

Abstract

Applications of Digital Image processing have increased exponentially in the recent years. Consequently, standards for the efficient representation and interchange of digital image are essential. Joint Photographic Experts Group (JPEG) is the most successful still image compression standard for bandwidth conservation. Basically a JPEG system consists of a Transformation unit followed by Normalizer and Encoder unit under image encoding. Under the Decoding side the Data is retrieved using the reverse operation to the Encoding. During the transmission it is a common approach of compressing the data before transmission so as to increase the rate of transmission with least bandwidth. This result in decompress the data increases resulting in slower operational speed. To achieve a higher compression and faster decoding time in this work an hybrid approach of variable and fixed length coding scheme is to be developed for the improvement of speed and compression ratio for digital image compression. To develop the suggested approach a MATLAB modeling of the JPEG coding system based on Huffman coding and the proposing hybrid coding is to be developed with the quality evaluation of Peak Signal to Noise Ratio over variable data rate and image representation.

Key words: *Hybrid, Huffman Coding, compression, transmission*

Introduction

Data compression has an important application in the areas of data transmission and data storage. Data compression is the process of converting data files into smaller files for efficiency of storage and transmission. As one of the enabling technologies of the multimedia revolution, data compression is a key to rapid progress being made in information technology. It would not be practical to put images, audio, and video alone on websites without compression. Compression is the process of representing information in a compact form. Data compression treats information in digital form that is, as binary numbers represented by bytes of data with very large data sets. Fox example, a single small 4 × 4 Size color picture, scanned at 300 dots per inch (dpi) with 24 bits/pixel of true color, will produce a file containing more than 4 megabytes of data. At least three floppy disks are required to store such a picture. This picture requires more than one minute for transmission by a typical transmission line (64k bit/second ISDN). That is why large data files remain a major bottleneck in a distributed environment. Although increasing the bandwidth is a possible solution, the relatively high cost makes this less attractive. Therefore, compression is a necessary and essential method for creating image files with manageable and transmittable sizes. In order to be useful, a compression algorithm has a corresponding decompression algorithm that, given the compressed file, reproduces the original file. There have been many types of compression algorithms developed. These algorithms fall into two broad types, lossless algorithms and lossy algorithms.

In many imaging applications, exact reproduction of the image bits is not necessary. In this case, one can perturb the image slightly to obtain a shorter representation. If this perturbation is much smaller than the blurring and noise introduced in the formation of the image in the first place, there is no point in using the more accurate representation. Such a code procedure, where perturbations reduce storage requirements is known as Image lossy coding. An uncompressed simple 512 x 512 grayscale image with 8 bits per pixel requires 262 kilobytes. To transmit the same using a 28.8K Modem requires a bandwidth of 2.1 Mega bits and the takes a transmission time of 1 min and 13 seconds. An uncompressed full motion 640 x 480, 1min (30 frames/sec) video with 24 bits per pixel requires 1.66 gigabytes. To transmit the same using a 28.8 K Modem requires a bandwidth of 221Mega bits and takes a transmission time of 5days and 8 hours approximately. The above example clearly illustrates the need for sufficient storage space, large transmission bandwidth, and long transmission time for image data. At the present state of technology, the only solution is to compress multimedia data before its storage and transmission, and decompress it at the receiver.

2. Basic Architecture Of Image Compression

A general system model for compression and decompression is:

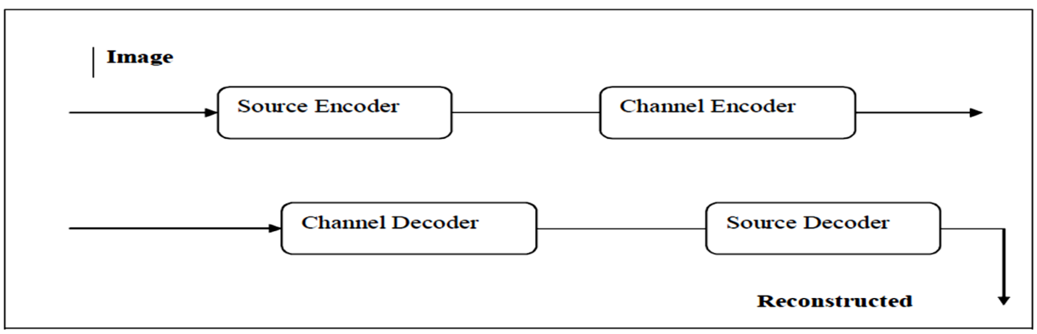


Figure 2.1 Block Diagram of an Image compression system.

Above Figure, shows Image compression system consists of two distinct structural blocks: an Encoder and Decoder. An input image is fed to encoder, which created a set of symbols from the input data. After transmission over the channel, the encoded representation is fed to the decoder where a reconstructed output image is generated. In general, the reconstructed image may or may not be replica of original image. If it is, the system is error free if not some level of distortion is present in the reconstructed image.

From mathematical point of view, the amount to transforming a 2D pixel array in to a statistically uncorrelated data set is known as encoding. The transformation is applied prior to storage or transmission of image and later the compressed image is decompressed to reconstruct the original image or an approximation to it.

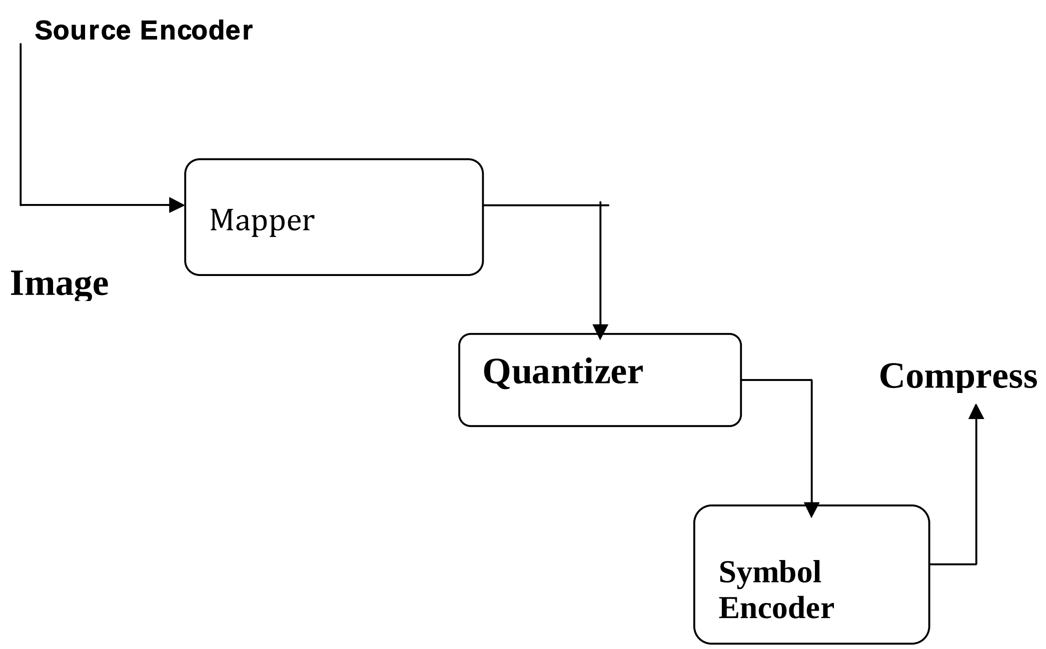


Figure 2.2 Block diagram of a Source Encoder

The Block Diagram of source encoder is shown in Figure 3.2. Which is responsible for reducing or eliminating any redundancies in the input image. Mapper transforms the input image into a format designed to reduce inter pixel redundancy. The mapper's transformed image can be restored. Quantizer fixes the coefficients of the mapper's output to some pre-determined values, which does not affect the image for reconstruction. Symbol encoder assigns a variable length code to the Quantized coefficients. Here by assigning a short code word to high probable coefficients, coding redundancy can be removed.

Mapper Transformation is done in two domains Spatial Domain Method which operate directly on the pixels in the image, and the transform coding in which a reversible linear transform is used to map the image into a set of transform coding in which a reversible linear transform is used to map the image into a set of transform coefficients which are then Quantized and encoded.

Source Decoder

Encoded Image

(compressed)

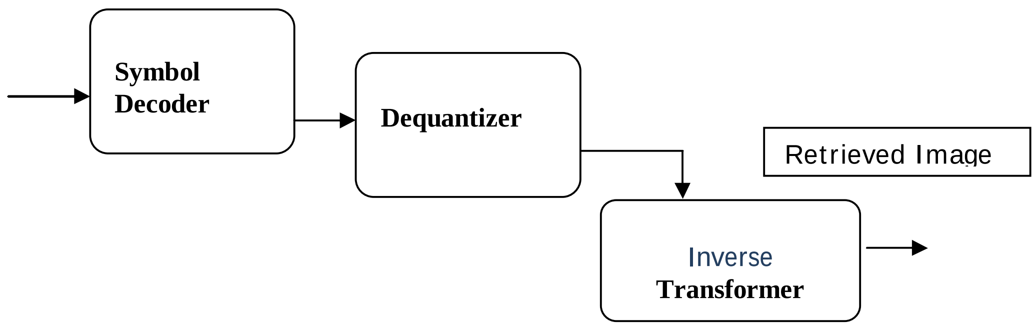


Figure 2.3. Block diagram of a Source decoder

Figure 3.3 shows the decoder system used for the decoding of the encoded compressed image data. The compressed data bit are passed to the symbol decoder where the encoded bits are decompressed following decompression algorithm used on the encoding side. The decompressed data is then passed to the inverse quantizer unit where the reverse quantization is carried out on the obtained data bits. Further the data bits are inverse transformed using the inverse transformation algorithm for the retrieval of the image bit.

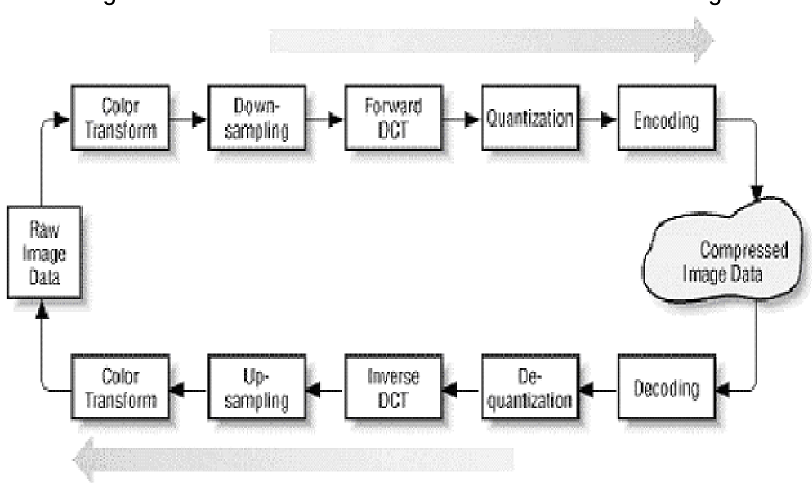


Figure 2.4 Overall basic architecture of image compression

The conventional architecture for image coding is as shown in figure 3.4, the system performs DCT transformation and entropy coding for image compression. An inverse process is performed over the compressed data to regenerate the original image back. During data storage or transportation the need for higher rate compression approach is required which could transform the information bit to lower representative bits and as well perform decoding operation in faster rate than the conventional decode operation. The realization of such a compression approach and the conventional approach is outlined in following chapter.

3.Image Compression Hybrid Coding Technique

3.1 Selective Coding

The compression/decompression scheme described in this project is based on Selective coding. In Selective coding, variable length code words are used to represent fixed-length blocks of bits in a data set. It is an extension for the Huffman coding algorithm known as Selective Huffman coding. For example, if a data set is divided into four-bit blocks, then there are 16 unique four-bit blocks. Each of the 16 possible four-bit blocks can be represented by a binary codeword. The size of each codeword is variable (it need not be four bits). The idea is to make the code words that occur most frequently have a smaller number of bits, and those that occur least frequently to have a larger number of bits. This minimizes the average length of a codeword. The goal is to obtain a coded representation of the original data set that has the smallest number of bits. A Huffman code is an optimal Selective code that is proven to provide the shortest average codeword length among all uniquely decodable variable length codes.

A Huffman code is obtained by constructing a Huffman tree. The path from the root to each leaf gives the codeword for the binary string corresponding to the leaf. An example of constructing a Huffman code can be seen in Table 4.3 for Figure 4.2. An example of a pixel set divided into four-bit blocks is shown in Figure 4.2, Table 4.3 shows the frequency of occurrence of each of the possible blocks (referred to as symbols). There are a total of 96 four-bit blocks in the example in Table 4.3.

An important property of Huffman codes is that they are prefix-free. No codeword is a prefix of another codeword. This greatly simplifies the decoding process. The decoder can instantaneously recognize the end of a codeword uniquely without any look ahead. Table 4.3 below shows the Selective Coding Based on Symbol Frequencies for Pixel Set given in Figure 4.2.

Using a Huffman code would provide the maximum compression; however, it would require a complex decoder and may not satisfy the constraint on the minimum size of a codeword. Therefore, some alternative Selective code must be selected. The approach taken here involves using a selective coding approach for which a very simple decoder can be constructed. Consider the case where the pixel set is divided into fixed-length blocks of b bits. There will be 2^b Code words. The first bit of each codeword will be used to indicate whether the following bits are coded or not. If the first bit of the codeword is a 0, then the next b bits are not coded and can simply be passed through the decoder as is (hence, the complete codeword has b + 1 bits). If the first bit of the codeword is a 1, then the next variable numbers of bits form a prefix-free code that will be translated by the decoder into a b-bit block. The idea is to only code the most frequently occurring b-bit blocks using code words with small numbers of bits (less than b, but greater than or equal to Lmin). Compression is achieved by having the most common b-bit blocks be represented by code words with less than b bits. The decoder is simple because only a small number of blocks are coded. The vast majority of the blocks is not coded and can be simply passed through the decoder. If n blocks are coded, then the decoder can be implemented with an FSM having no more than n + b states (compared with a Huffman code which requires $2^b - 1$ states).

Table 4.3 Selective Coding based on Symbol Frequencies

I	x_i	Occ	Pi	Huffman code	Selective code
1	1111	30	0.3125	11	**1**0
2	1011	12	0.1250	100	**1**10
3	0111	10	0.1042	010	**1**11
4	1101	09	0.0938	000	**0**1101
5	1110	08	0.0833	1011	**0**1110
6	0110	05	0.0521	0011	**0**0110
7	0000	04	0.0417	10101	**0**0000
8	0010	04	0.0417	01111	**0**0010
9	1001	03	0.0313	01110	**0**1001
10	0101	03	0.0313	01101	**0**0101
11	1100	02	0.0208	01100	**0**1100
12	0100	02	0.0208	00101	**0**0100
13	1000	02	0.0208	00100	**0**1000
14	1010	02	0.0208	101001	**0**1010
15	0011	0	0	101000	**0**0011
16	0001	0	0	UUUUU	UUUUU
# states of FSM				15	7

Table.4.4 Example of pixel set divided into four-bit blocks

1111	0101	0011	1111	1011	1110	1101	1011
1111	1111	1111	1111	0000	0000	0000	0000
1001	0010	0110	0111	1110	1101	0110	1110
1111	1111	1110	0110	1111	1001	1111	1011
1111	1111	0111	1010	1111	1111	0111	1111
0010	1110	1000	0100	1111	1111	1111	1111
0110	1011	1011	1011	1101	0111	1011	0111
1100	1000	1010	0111	0101	1011	1111	1101
1011	0100	1101	1101	1110	1111	1111	1111
1111	1011	0111	0101	1111	0111	1111	1101
1100	1101	1001	1110	1110	1101	1011	0110
0111	0010	1111	1111	0111	1111	1011	1111

4. Experimental Results

4.1 Input Consideration

For the analysis of the proposed system various images of .tiff format were read and processed. The image is processed using the conventional JPEG codec architecture, wherein Huffman based entropy coder is used. The obtained simulation results are as outlined below,

(a)

(b)

Figure 4.1(a) Original image sample; (b) Retrieved image after processing

The obtained evaluative metric for the processed image based on Huffman based coder is observed as;

Compression Rate 1.6875 Bits / pixel
Encoding Time Taken 1.7813
Decoding Time Taken 4.6406
Total processing Time Taken 6.4219
SNR = 31.7087

A similar operation is carried out for the same image considered in conventional JPEG coder with the proposed Selective based image coder. The obtained results for such a system is presented below,

(a)

(b)

Figure 4.2(a) Original Image; (b) Retrieved image after processing

International Journal of Academic Research
ISSN: 2348-7666 Vol.2, Issue-1(4), January-March, 2015

The observations obtained for the developed system based on selective decoder is as presented below,

Compression Rate 2.4531 Bits / pixel

Encoding Time Taken 1.4375

decoding Time Taken 1.9219

Total processing Time Taken 3.3594

SNR = 31.7087

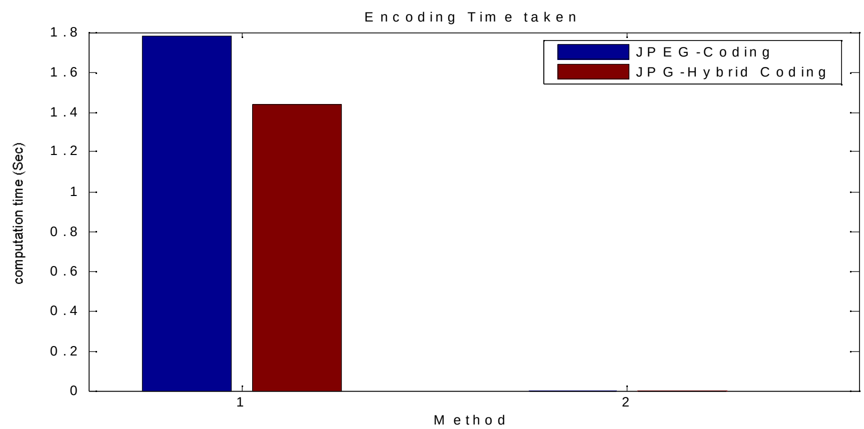


Figure 4.3 Computation Time plot for encoding operation for the two method

From the obtained observation above it cane observed that in case of JPEG based coder the total computation time for encoding taken is 1.8 second whereas when compare to JPEG coder with selective coding take nearly 1.4 seconds to encode the same data. This results in a reduction of 0.4 seconds system computation delay resulting in faster operation.

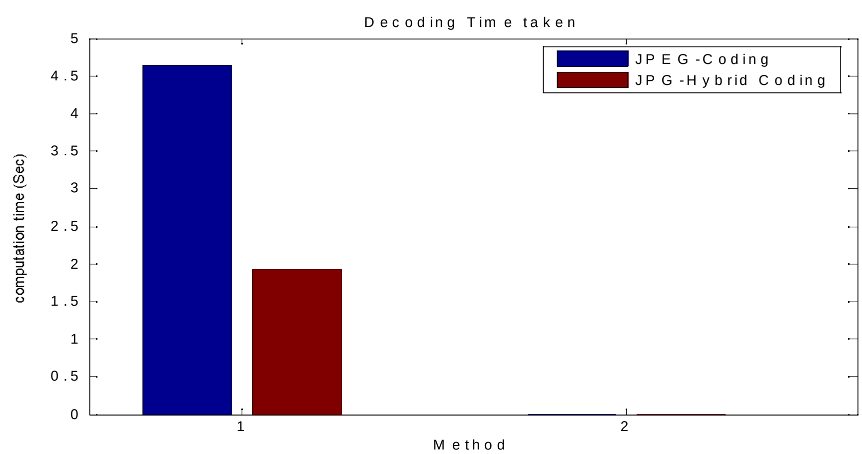


Figure4.4 system computation time for performing decoding operation

The obtained observation of total computation time for decoding operation is shown in figure 4.4. It could be observed that the conventional JPEG coder with Huffman based coding technique consumes about 4.6 seconds for decoding the encoded data. In comparison the decoder unit performs the same operation in about 1.8 seconds of time. This leads a reduction of total decoding time for about 2.8 seconds or about 50~60% in computation time.

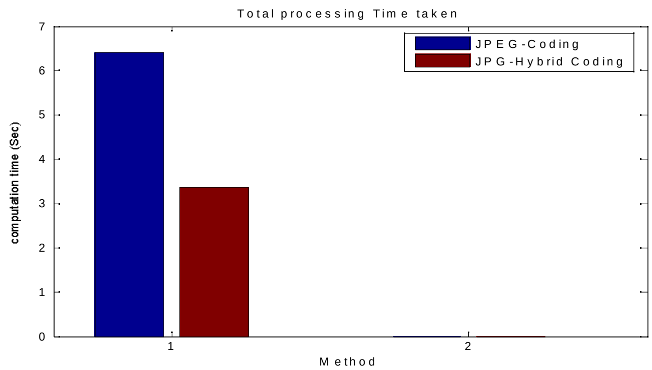


Figure 4.5 Total system delay for the developed method

Figure illustrates the total processing time comparison between JPEG and Selective Coding based coders for processing the given test image. A speed improvement of about 50 % is observed for the suggested system as compared to the conventional approach.

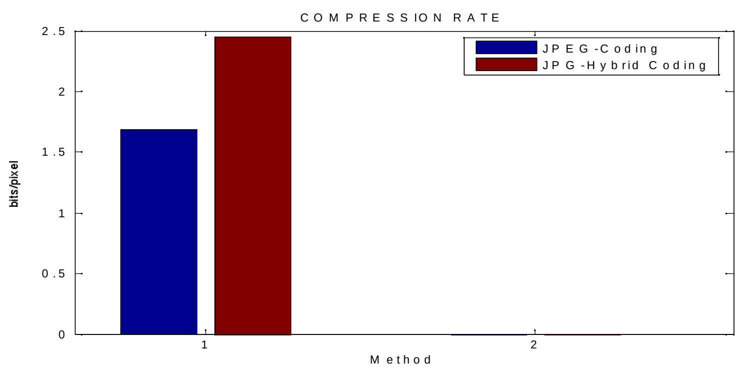
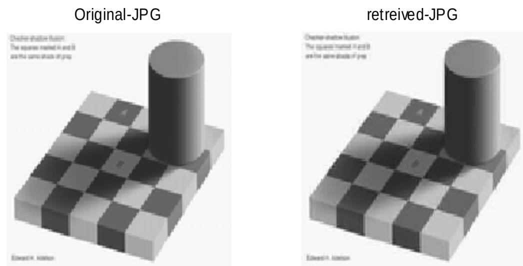


Figure 4.6 compression rate for the two developed method

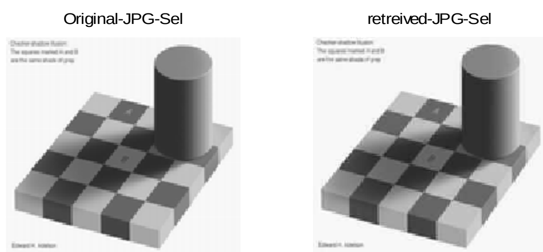
Figure 4.6 illustrates a comparison of the compression rate achieved for the two methods. The bit/pixel representation for conventional coding is obtained to a limit of 1.7, whereas in case of selective coder the limit obtained is 2.48. There is about 0.7 bit/pixel improvement is observed for the given sample.

A similar observation is carried out on other samples and the obtained results were presented below,



(a) (b)

Figure 4.7 (a) original image (b) retrieved image after JPEG coding



(a) (b)

Figure 4.8 (a) original image (b) retrieved image after JPEG-Selective coding

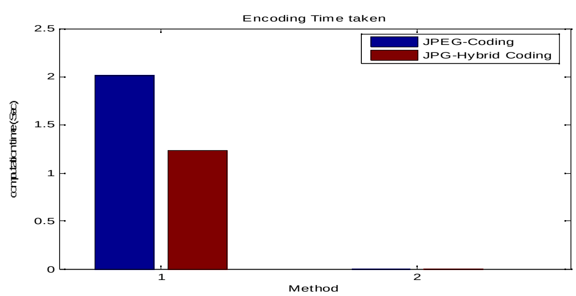


Figure 4.9 Encoding time computation plot for the two systems

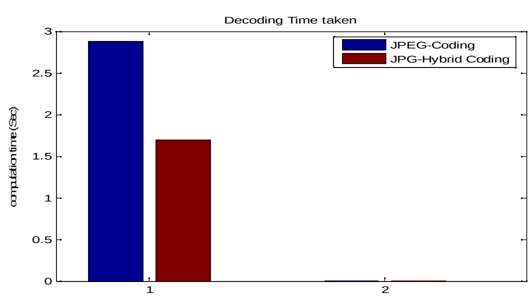


Figure 4.10 Decoding time computation plot for the two systems

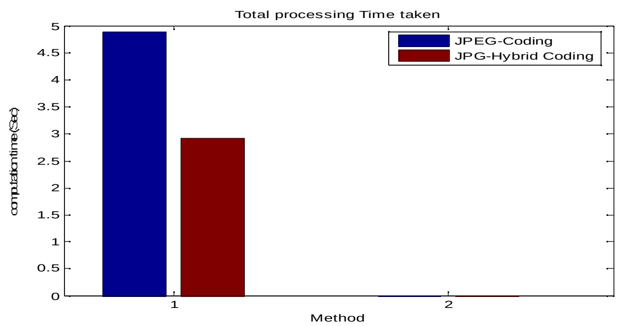


Figure 4.11 overall system time computation plot for the two systems

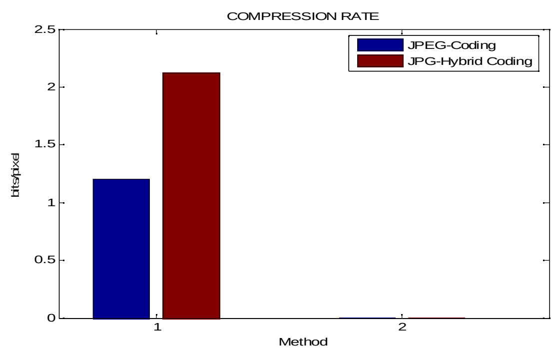


Figure 4.12 Bit/pixel for the two systems

5. Conclusion

In this work a selective coding approach is developed for the conventional JPEG coders who incorporate the property of both VLC and FLC coding for image compression. The suggested approach is observed to improve the performance to about 50~60% in system computation time and about 30~35% of bit/pixel as compared to the conventional JPEG coding. The developed system is hence useful to the environment where the resources with regard to bandwidth and power are constraints. The lower in computation time can reduce the power consumption and also the faster system can result in higher throughput even under constraint resource conditions.

A future analysis of channel condition with real time device architectures could be carried out for the suggested approach for the evaluation of system feasibility and quality improvement of the processed data in real time resource constraint environment.

References

[1]R.C.Gonzalez & R.E.Woods, 'Digital Image Processing', Addison Wesley, ISBN: 0-201-60078.

[2] "Joint Photographic Experts Group, JPEG ". 2009-11-08.

[3] Information technology - Digital compression and coding of continuous-tone still images - Requirements and guidelines" 2009-11-07.

[4] Subhasis Saha "Image Compression - from DCT to Wavelets : A Review"ACM Crossroads Student Magazine The ACM's First Electronic Publication.

[5]Rudra Pratap " Getting Started With Matlab: A Quick Introduction for Scientist And Engineer" version 6 Oxford University press, 2003.

[6] D.A. Huffman, "A Method for the Construction of Minimum Redundancy Codes," Proceedings of the IRE, Vol. 40, No. 9, September 1952, pp. 1098-1101.

[7] J. Ziv and A. Lempel, "A Universal Algorithm for Sequential Data Compression," IEEE Trans. on Information Theory, Vol. IT-23, May 1977, pp. 337-343.

[8] J. Ziv and A. Lempel, "A Compression of Individual Sequences via Variable-Rate Coding," IEEE Trans. On Information Theory, Vol. IT-24, September 1978, pp. 530- 538.

[9] T.A. Welch, "A Technique for High-Performance Data Compression," IEEE Computer, Vol. 17, No. 6, June 1984, pp. 8-19.

[10] S. Apiki, "Lossless Data Compression," Byte, Vol. 16, No. 3, March 1991, pp. 309-312, 314, 386-387.