International Journal of Computer Science
and Engineering (IJCSE)
ISSN(P): 2278-9960; ISSN(E): 2278-9979
Vol. 3, Issue 4, July 2014, 11-18

International Academy of Science,
Engineering and Technology
Connecting Researchers; Nurturing Innovations

IMPLEMENTATION OF IMAGE COMPRESSION USING MODIFIED SPIHT ALGORITHM FOR SPACE APPLICATIONS

MANIMEGALAI V

KPR Institute of Engineering and Technology, Kollupalayam, Tamil Nadu, India

ABSTRACT

This paper describes a well known algorithm of Pearlman Set Partitioning in Hierarchical Trees SPIHT is to restrict the necessity of random access to the whole image to a small sub images only. The main idea is based on partitioning of sets, which consists of coefficients or representatives of whole sub trees. The decoder duplicates the execution path of the encoder to ensure this behavior; the coder sends the result of a binary decision to the decoder before a branch is taken in the algorithm. Thus, all decisions of the decoder are based on the received bits. The name of the algorithm is composed of the words *set* and partitioning. The compression performance is measured with peak signal to noise ratio compared to the original codec remains still the same or nearly the same. This code can be implemented through MATLAB.

KEYWORDS: Image Codec, Image Compression, Digital Image Processing

INTRODUCTION

Image compression is the art and science of reducing the amount of data required to represent an image, is one of the most useful and commercially successful technologies in the field of digital image processing. The numbers of images that are compressed and decompressed daily is staggering and the compression and decompression themselves are virtually invisible to the user. The image compression of two types, they are lossy and lossless. In lossy compression the image compression would reconstruct the image with a varying degree of information loss. In lossless the reconstructed image is exactly the same as the. Original one without any information lost. An image compression system is composed of two distinct functional components are an encoder and a decoder. The encoder performs compression and the decoder performs the complementary operation of decompression. Both operations can be performed in software, as in the case in web browsers and many commercial image editing programs, or in a combination of hardware and finware, as in commercial DVD players. A codec is a device or program that is capable of both encoding and decoding.

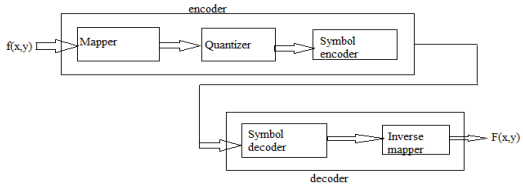

Figure 1: Functional Block of General Image Compression System

The encoder is designed to remove the redundancies through a series of three independent operations. In the first stage of the encoding process, a mapper transforms f(x...) into a non-visual forma designed to reduce spatial and temporary redundancy. This operation generally reversible and may or may not reduce directly the amount of data required to

represent the image. Run length-coding is an example of mapping that normally yields compression in the first step of encoding process. The mapping of an image into a set of less correlated transform coefficients is an example of the opposite video applications. The mapper uses previous video frames to facilitate the removal of temporal redundancy. The quantizer reduces the accuracy of the mappers output in accordance with a pre-established fidelity criterion. The goal is to keep irrelevant information out of compressed representation. This operation is irreversible. It must be omitted when error-free compression is desired. In video applications, the bit rate of the encoded output is often measured and used to adjust the operation of the quantizer so that a predetermined average output rate is maintained. Thus the visual quality of the output can vary from frame to frame as a function of image content. In the third and final stage of the encoding process, the symbol coder generates a fixed or variable length code to represent the quantizer output and maps the output in accordance with the code. In many cases a variable length code is used. The shortest code words are assigned to the most frequently occurring quantizer output values thus minimizing coding redundancy. This operation is reversible. Upon its completion, the input image has been processed for the removal of each of the three redundancies.

The decoder contains only one two components a symbol decoder and an inverse mapper. They perform in reverse order, the inverse operation of the encoder's symbol encoder and mapper. Because quantization results in irreversible information loss, an inverse quantizer block is not included in the general decoder model. In video applications, the decoded output frames are maintained in an internal frame store and used to reinsert the temporal redundancy that was removed at the encoder.

SPIHT ALGORITHM

In SPIHT algorithm, the image is decomposed into a number of sub bands by means of hierarchical wavelet decomposition. The sub band coefficients are grouped into sets as spatial-orientation trees, which efficiently exploit the correlation between frequency bands. The coefficients in each spatial orientation tree are progressively coded from the most significant bit-planes (MSB) to the least significant bit-planes (LSB), starting the coefficients with highest magnitude and at the lowest pyramid levels.

The SPIHT Multistage Encoding Process Employs Three Lists and Sets

- The list of insignificant pixels (LIP) contains individual coefficients that have magnitudes smaller than the threshold.
- The list of insignificant sets (LIS) contains sets of wavelet coefficients that are defined by tree structures and are found to have magnitudes smaller than the threshold (insignificant). The sets exclude the coefficients corresponding to the tree and all sub tree roots and they have at least four elements.
- The list of significant pixels (LSP) is a list of pixels found to have magnitudes larger than the threshold (significant).
- The set of offspring (direct descendants) of a tree node, O (i, j), in the tree structures is defined by pixel location (i, j). The set of descendants, D (i, j), of anode is defined by pixel location (i, j). L (i, j) is defined as $L(i, j) = D(i, j) – O(i, j)$..

Procedure

Step 1: Initialization

n = [log2 (max |coeff|)]

LIP = All elements in H

LSP = Empty

LIS = D's of Roots

Step 2: Significance Map Encoding ("Sorting Pass")

Process LIP

For each coeff (i,j) in LIP

Output Sn (i, j)

If Sn (i, j) =1

Output sign of coeff (i, j): 0/1 = -/+

Move (i,j) to the LSP

End if

End loop over LIP

Process LIS

For each set (i, j) in LIS

If type D

Send Sn (D (i, j))

If Sn (D (i, j)) =1

For each (k, l) $\in$ O (i, j)

Output Sn (k, l)

If Sn (k, l) =1, then add (k, l) to the LSP and output sign of coeff: 0/1 = -/+

If Sn (k, l) =0, then add (k, l) to the end of the LIP

End for

End if

Else (type L**)**

Send Sn (L (i, j))

If Sn (L (i, j)) =1

Add each (k, l) $\in$ O (i,j) to the end of the LIS as an entry of type D

Remove (i, j) from the LIS

End if on type

End loop over LIS

Step 3: Refinement Pass: for each entry in the SP, except those included in the last process for sorting, output the nth most significant bit of |i, j|;

Step 4: Loop: reduced n by 1 and go to X if needed.

MODIFIED SPIHT ALGORITHM

Most definitions of the lists and symbols are identical to the original SPIHT except that 1-D addresses are used instead of 2-D addresses and three definitions, that is, MaxLIP, MaxLIS and MaxLSP, are increased. They are as follows:

O (i): Set of coordinates of all off spring of node i

D (i): Set of coordinates of all descendants of node i

H: Set of coordinates of all spatial orientation tree roots

L (i): D (i) − O (i)

Ci: The magnitude of the coefficient of node i

MaxLIP: The maximal address of all nodes in the LIP list

MaxLIS: The maximal address of all nodes in the LIS list

MaxLSP: The maximal address of all nodes in the LSP list

T: All nodes in the image.

N: The address of the last node in the image.

A set L (i) or D (i) is said to be significant if any coeffi-cient in the set has a magnitude greater than the thresh-old, such as

Sn (G) =1, max {| ci |},∀(i) ∈ G

0, otherwise.

Where G, G ⊆ T, is a set of nodes.

Similar to the SPIHT algorithm, four encoding steps, initialization, sorting pass, refinement pass and quantization pass, are performed and three linked lists, LIS, LSP and LIP are used in the proposed SPIHT. Its pseudo code is described as follows:

Initialization

Output n = _log2 (max {| ci |}) _, 0 ≤ i ≤ N;

Set LSP (i) = 0, 0 ≤ i ≤ N;

Set LIP (i) = 1,∀i ∈ H, otherwise set to 0;

Set LIS (i) = A,∀i ∈ H with descendants, otherwise set to 0;

Set MaxLIP = max (H), MaxLIS = max (H), MaxLS P = 0;

Refinement Pass

For 0 ≤ i ≤ MaxLS P

If LS P (i) = 1 then

Output the nth most significant bit of | ci |;

Else if LS P (i) = 2 then Set LS P (i) = 1;

Sorting Pass

For 0 ≤ i ≤ MaxLIP

If LIP (i) = 1 then

Output S n ((i)),

If S n ((i)) = 1, then

Output sign of ci

Set LSP (i) = 2,

MaxLSP = max (i, MaxLSP)

Set LIP (i) = 0

For 0 ≤ i ≤ MaxLIS

If LIS (i) = A then

Output Sn (D (i))

If Sn (D (i)) = 1 then

For (i × 4) ≤ j ≤ (i × 4 + 3)

Output Sn ((j))

If Sn ((j)) = 1 then Set LSP (j) = 2,

MaxLSP = max (i, MaxLSP) Output the sign of c j

Else

Set LIP (j) = 1,

MaxLIP = max (i, MaxLIP)

If (i × 16) < N then Set LIS (i) = B,

MaxLIS = max (i, MaxLIS)

Else

Set LIS (i) = 0

If LIS (i) = B then

Output Sn (L (i))

If Sn (L (i)) = 1, then

For $(i \times 4) \leq j \leq (i \times 4 + 3)$

Set LIS (j) = A

Set LIS (i) = 0

Quantization-step update pass
Decrement n by one and go to Step 2.

In both SPIHT algorithms, there are three lists, LIS, LIP, and LSP to be constructed. When an element in LIS changes its type, transactions among the lists are necessary. Insertion and deletion operations of the lists are required, too. Dynamic linked lists have to be employed to construct the three lists for the SPIHT algorithm. It is easy to construct dynamic linked lists with high level programming languages because memory management mechanism is avail-able in computer operating systems. However, design of such a dedicated low cost circuit is not so straightforward. These operations reduce the throughput at the same time. With the proposed approach, no dynamic linked list is necessary. If N is the total number of coefficients, based on the proposed addressing method, addresses of the coefficients which have descendants are from 0 to (N/4) −1. Tables with N/4 entries are used for the lists. They are LIS (i), LIP (i), and LIS (i), for i = 0... (N/4) − 1. LIS (i) is set to be zero when node-i has no descendant, or it has not joined the encoding process yet. Otherwise, it denotes a coefficient at position-i as either type A or type B. Similar definitions are used for LIP (i) and can be found in the algorithm. LSP (i) is set to be zero when node-i is insignificant pixel. LSP (i) is set to be 2 when node-i becomes significant pixel for the first time, then it is set to be 1 after the node-i passes the refinement pass. By scanning the tables subsequently, one encoding pass is finished. Simple counters and finite state machines (FSM) are enough for its implementation. The test images are 8 bpp grey scale images. Bi-orthogonal (9, 7) filter and 5-level DWT are used. The other pre-processing follows the suggestions in JPEG200.

CONCLUSIONS

In this work, we have presented an efficient hardware for image codec based on modified SPIHT. Efficient addressing method for accessing error matrix elements (coefficients) is used in the implementation. The hardware can be realized at low cost. The distortion measure will be the bottleneck for such technique at lower code rates, which will be addressed in system architecture. The hardware design is verified over Xilinx Vertex device. The clock frequency (speed) and throughput are increased to a greater extent with 1D addressing scheme as compared to that of 2D addressing scheme. The speed is increased by 66.67% and throughput is increased by 61.11%. In the future much more effort must be emerged in order to make the codec more resilient against bit or synchronization errors, which should be quite a challenging task in order to keep the embedded character of the bit stream together with the progressive behavior.

REFERENCES

1. Said and W. A. Pearlman, “A new fast and efficient image codec based on set partitioning in hierarchical trees,” IEEE Trans. Circuits Syst. Video Technol., vol. 6, no. 12, pp. 243-250, June 1996
2. D. Taubman, “High Performance scalable image compression with EBCOT,” IEEE Trans. Image Process., vol. 9, no. 7, pp. 1158-1170, July 2000.
3. P. Luigi Dragotti, G. Poggi, A. R. P. Ragozini, Compression of multispectral images by three-dimensional SPIHT algorithm, IEEE Trans on Geoscience and Remote Sensing 38(1) (2000) 416-461.
4. C. Lambert-Nebout, G Y. Mourof, A survey of on-board image compression for CNES space missions, in: proceddings of the 1999 international geosciences and remote sensing symposium, 1999.
5. E. Delp, O. Mitchell, "Image compression using block truncation coding,” IEEE Trans on communications 1979.
6. M. Rabbani and P. W. Hones., digital image compression techniques, SPIE Opt, eng, 1991.
7. M. Antonini, M. Barlaud P. Mathieu and I. Daubechies, "Image coding using wavelet transforms." IEEE Trans, April 1992.
8. J. M. Shapiro, “Embedded image coding using zero tree of wavelet coefficients.” IEEE Trans, Dec 1993.