ELectronics EXpress

IEICE Electronics Express, Vol.1, No.15, 447–452

Improved EZBC algorithm with low complexity

Wei-na Du, Jun Sun, and Miao Sima

Institute of Image Communication & Information Processing, Shanghai Jiaotong University

1954 Huashan Road, Shanghai 200030, People's Republic of China

Abstract: In this paper, an improved algorithm based on EZBC and its implementation architecture with low complexity and high performance are presented. The proposed algorithm utilizes the significance state table of quadtree nodes and the context look-up table to control the coding passes and form the context, so the proposed algorithm requires low memory and has low implementation complexity with the nearly same performance.

Keywords: zeroblock, quadtree, embedded coding

Classification: Science and engineering for electronics

References

[1] A. 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. 3, pp. 243–249, 1996.

[2] D. Taubman, "High performance scalable image compression with EBCOT," *IEEE Trans. Image Processing*, vol. 9, no. 7, pp. 1158–1170, 2000.

[3] M. Rabbani and R. Joshi, "An overview of the JPEG 2000 still image compression standard," *Signal Processing: Image Communication*, vol. 17, no. 1, pp. 3–48, 2002.

[4] S.-T. Hsiang and J. W. Woods, "Embedded image coding using zeroblocks of subband/wavelet coefficients and context modeling," *Proc. IEEE ISCAS'00*, vol. 3, pp. 662–665, Geneva, Switzerland, May 2000.

[5] S.-T. Hsiang, "Highly scalable subband/wavelet image and video coding," Ph.D. Thesis, Rensselaer Polytechnic Institute, 2002.

1 Introduction

Zerotree-based codec and zeroblock-based codec of the subband/wavelet coefficients, two popular embedded coding techniques, take full advantages of the nature of energy clustering of subband/wavelet coefficients in frequency and in space. SPIHT [1], a popular zerotree-based codec, defines set partitioning in hierarchical trees during encoding or decoding procedure. The remarkable zeroblock-based approach EBCOT [2], is the core compression algorithm of international standard JPEG2000 [3]. The zerotree-based codec mainly uses the correlation of the inter-band, while the zeroblock-based scheme generally

DOI: 10.1587/elex.1.447
Received August 27, 2004
Accepted October 12, 2004
Published November 10, 2004

IEICE Electronics Express, Vol.1, No.15, 447–452

utilizes the correlation of the intra-band.

An embedded image coding algorithm using zeroblocks of subband/wavelet coefficients and context modeling (EZBC) [4] was proposed by Shih-Ta Hsiang. EZBC combines the advantages of zeroblock coding and context modeling of the subband/wavelet coefficients by fully utilizing the correlation of both inter-band and intra-band. Thus EZBC outperforms SPHIT and can be competitive with EBCOT in compression efficiency. But a large amount of memory is required to maintain two lists that are used to store the coordinates of the quadtree nodes needed to be coded in the bitplane coding procedure, also a great amount of operations to read and write the memory are required in each coding pass. These become the main drawbacks for hardware implementation.

A simpler image coding algorithm based on EZBC is presented in this paper, with low complexity and high performance. The proposed algorithm utilizes the significance state-table of the quadtree nodes forming the context modeling to control the coding passes instead of the two lists in the original EZBC, and achieves the nearly same performance with low memory requirement and low complexity that making it hardware implementation friendly.

2 Proposed algorithm based on EZBC

In EZBC, the coding process begins with establishment of the quadtree representation for the individual subband [4, 5]. The bottom quadtree level named as pixel level is made up of the magnitude of every subband coefficients. Each quadtree node of the next higher level is set to the maximum value of the four corresponding nodes at the current level. From the most significant bit (MSB) toward the least significant bit (LSB), EZBC progressively encodes the bitplane of subband coefficients. Whenever a node tests significant against the current threshold, it is split into four descendent nodes. This testing and splitting procedure is recursively performed until the bottom level [5]. EZBC adopts the significance status of neighbor nodes and parent subband node to form the context models. This complicated context modeling scheme can encode this quadtree structure efficiently.

EZBC also uses the list of the insignificant nodes (LIN) and the list significant pixels (LSP) to control the bitplane coding passes. The nodes or pixels are added into the corresponding list according to their significance states, while in the next coding pass the coder encodes these nodes or pixels in the lists directly. So a large amount of memory is required in this coder for the two lists to maintain these coordinates of nodes or pixels. In our scheme, we establish a significance state-table to identify whether the nodes or pixels need to be coded rather than maintain their coordinates and form the context simultaneously. The main contribution of our work is the utilization of the significance status in forming context procedure to control the coding passes, eliminating the two lists. The proposed approach is of low memory requirement and fixed processing operations.

DOI: 10.1587/elex.1.447
Received August 27, 2004
Accepted October 12, 2004
Published November 10, 2004

The proposed algorithm is outlined as follows.

2.1 Definition

$c_k(i,j)$: the quantized subband coefficient of subband k at position (i,j).

$Q_k[l](i,j)$: the quadtree node representation at position (i,j), subband k and level l.

$$Q_k[l]\,(i,j) \equiv \begin{cases} |c_k(i,j)| & , \; l=0 \\ \max\left\{ \begin{array}{l} Q_k[l-1](2i,2j), \\ Q_k[l-1](2i+1,2j), \\ Q_k[l-1](2i,2j+1), \\ Q_k[l-1](2i+1,2j+1) \end{array} \right\} & , \; l>0 \end{cases}$$

D_k : the depth of the quadtree of the subband k.

$D_{\max}$: $\max_{\{k\}}\{D_k\}$.

K : total number of subbands.

$S_n(i,j)$: the function to test the significance against the threshold of bitplane n.

$$S_n(i,j) = \begin{cases} 1 & , \; Q_k[l](i,j) \geq 2^n \\ 0 & , \; otherwise \end{cases}$$

$\sigma_k[l](i,j)$: the significance state-table of node $Q_k[l](i,j)$.

$$\sigma_k\,[l]\,(i,j) = \begin{cases} 1 & , \; significant \\ 0 & , \; insignificant \end{cases}$$

2.2 Coding passes

1) Initialization

$$\sigma_k = 0$$
$$n = \left\lfloor \log_2(\max_{(k,i,j)}\{|c_k(i,j)|\}) \right\rfloor$$

2) for $l = 0 : D_{\max}$

$$\text{for } k = 0 : K-1$$
$$CodeInsignificantNode(k,l)$$

3) for $k = 0 : K-1$

$$CodeSignificantPixel(k)$$

4) $n = n-1$ and go to 2)

2.3 Pseudo code

$CodeInsignificantNode(k,l)\{$

for each $\sigma_k[l](i,j) = 0$ and $\sigma_k[l+1](i/2,j/2) = 1$

code $S_n(i,j)$

if $(S_n(i,j) = 1)$

- set $\sigma_k[l](i,j) = 1$

DOI: 10.1587/elex.1.447
Received August 27, 2004
Accepted October 12, 2004
Published November 10, 2004

- if $(l = 0)$ code the sign bit of the $c_k\,(i, j)$
- else $CodeQuadTree(k, l, i, j)$

}

$CodeSignificantPixel(k)\{$

for each $\sigma_k[0](i, j) = 1$ and $|c_k\,(i, j)| \geq 2^{n+1}$

code the nth MSB of $c_k\,(i, j)$

}

$CodeQuadTree(k, l, i, j)\{$

for each $(x, y) \in \{(2i, 2j), (2i, 2j + 1), (2i + 1, 2j), (2i + 1, 2j + 1)\}$ of the level $l - 1$

code $S_n(x, y)$;

if $(S_n\,(x, y) = 1)$

- set $\sigma_k[l - 1]\,(x, y) = 1$
- if $(l = 1)$ code the sign bit of $c_k\,(x, y)$
- else $CodeQuadTree(k, l - 1, x, y)$

}

3 Proposed architecture

The variable length lists and a lot of memory required make the hardware implementation of EZBC difficult. We bring forward an improved algorithm and a proposed architecture to overcome these drawbacks. The block diagram of the proposed encoder is shown in Fig. 1. In this figure, the quantized wavelet coefficients of subband are loaded into the proposed encoder. The process begins with input of the sign bits and establishment of the quadtree representation for the inputted subband coefficients. We use the state variable memory to store the significance state of the nodes or pixels that are needed for context formation and coding passes controlling. The roles of the state variable register are to load the required state variables, update the state variables and send the significance states of neighbor nodes or pixels at different orientations (H, V, D) [5], the inter-band dependency (P) and the significance (σ_{sb}) of past coding siblings to the corresponding processing element. Sign register sends the sign contributions at different orientations and sign bit to the sign coding element. Contexts corresponding to every

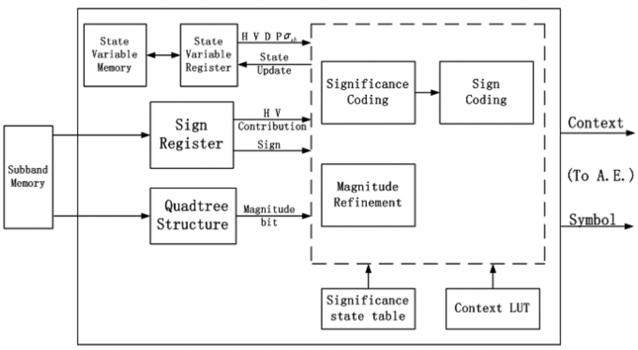

Fig. 1. Block diagram of proposed architecture.

© IEICE 2004
DOI: 10.1587/elex.1.447
Received August 27, 2004
Accepted October 12, 2004
Published November 10, 2004

ELectronics EXpress

symbol are formed through the lookup table (LUT) in the proposed architecture, which are the information needed for arithmetic encoder (A.E.). The details of the processing elements and coding pass controlling are described in Section 2.

4 Experimental results

The dyadic wavelet decomposition with Daubechies 9/7 filters is employed for the test. The context models utilized in EZBC and the proposed algorithm both are based on Ref. 5. Shown in Table I are the PSNR values of three standard test grayscale images of size 512 × 512, which are Lena, Barbara and Goldhill. The results reveal that the compression performance (PSNR) of this proposed algorithm is nearly same as EZBC.

Table I. PSNR (dB) comparison of proposed algorithm and EZBC

Image	Rate (bpp)	EZBC	Proposed
Lena	1.0	40.58	40.58
	0.5	37.44	37.44
	0.25	34.33	34.32
Barbara	1.0	37.15	37.17
	0.5	32.07	32.08
	0.25	28.18	28.20
Goldhill	1.0	36.78	36.78
	0.5	33.37	33.37
	0.25	30.72	30.71

EZBC requires a large and variable memory for the two lists (LIN, LSP) that increase the hardware complexity. EZBC also utilizes the significance status of neighbor nodes and parent subband node to form the context modeling. In this paper, we take full advantages of the significance state-table of nodes to form the context and control coding passes simultaneously, so we reduce memory greatly, comparison of the memory usage is shown in Table II at 1 bpp. Obviously, the proposed algorithm is of low memory requirement and low complexity, which results in a simpler implementation for hardware.

Table II. Memory usage comparison (bytes)

Codec	512 × 512	2048 × 2560
EZBC	190 K	4,400 K
Proposed	40 K	880 K

DOI: 10.1587/elex.1.447
Received August 27, 2004
Accepted October 12, 2004
Published November 10, 2004

5 Conclusion

An improved quadtree coding algorithm based on EZBC and its implement architecture are presented in this paper. The proposed algorithm exploits the significance state-table to not only control the coding passes but also form the context, thus achieving the nearly same compression performance and making it hardware implementation friendly as well.

DOI: 10.1587/elex.1.447
Received August 27, 2004
Accepted October 12, 2004
Published November 10, 2004