2011

Low-Complexity Image Coder/Decoder With An Approaching-Entropy Quad-Tree Search Code For Embedded Computing Platforms

Tao Ma
University of Nebraska Lincoln, tma@unlnotes.unl.edu

Pradhumna Shrestha
University of Nebraska Lincoln, plshrestha@unlnotes.unl.edu

Michael Hempel
University of Nebraska Lincoln, mhempel@unlnotes.unl.edu

Dongming Peng
University of Nebraska Lincoln, dpeng@unlnotes.unl.edu

Hamid Sharif
University of Nebraska-Lincoln, hsharif@unlnotes.unl.edu

2011 18th IEEE International Conference on Image Processing

Digital Object Identifier: 10.1109/ICIP.2011.6116287

LOW-COMPLEXITY IMAGE CODER/DECODER WITH AN APPROACHING-ENTROPY QUAD-TREE SEARCH CODE FOR EMBEDDED COMPUTING PLATFORMS

Tao Ma, Pradhumna Shrestha, Michael Hempel, Dongming Peng, Hamid Sharif
Department of Computer and Electronics Engineering, University of Nebraska Lincoln
{tma,plshrestha,mhempel,dpeng,hsharif}@unlnotes.unl.edu

ABSTRACT

In this paper, we propose a fast, simple and efficient image codec applicable for embedded processing systems. Among the existing image coding methods, wavelet quad-tree is a foundation leading to an efficient structure to encode images. By searching significant coefficients along quad-trees, an embedded efficient code can be obtained. In this work, we exploit hierarchical relations of the quad-tree structure in terms of searching entropy and present a quad-tree searching model that is very close to the searching entropy. By applying this model, our codec surpasses SPIHT [1] by 0.2-0.4 db over wide code rates, and its performance is comparable to SPIHT with arithmetic coding and JPEG2000 [2]. With no additional overhead of arithmetic coding, our code is much faster and simpler than SPIHT with adaptive arithmetic coding and the more complicated JPEG2000 algorithms. This is a critical factor sought in embedded processing in communication systems where energy consumption and speed are priority concerns. Our simulation results demonstrate that the proposed codec is about twice as fast with very low computational overheads and comparable coding performances than existing algorithms.

Index Terms—Image coding, quad-tree search model, embedded communication system.

1. INTRODUCTION

Image coding has been extensively studied by many researchers for several decades. Though there are many results reported for the general image coding algorithms [3] –[4], relatively few efforts have been conducted for the purpose of reducing computing overhead (e.g., execution time, runtime computation data storage requirements, computing complexities) in the image coding/decoding algorithms. Recent advances in embedded computing and communication systems and applications, such as the deployment of wireless multimedia sensor networks [5]–[7], necessitate low-complexity and fast image coding/decoding algorithms which are applicable to microcontroller and mini-processor based computing environment, where the systems are constrained by resource limitations such as computing and communications energy constraints, buffer size limitations and bandwidth limitations.

In this paper, we propose a low-complexity and high-speed image codec based on a new exploitation methodology in quad-tree structures to encode images with applicability to embedded systems for faster coding and especially faster reconstruction of the compressed images.

Because of its efficient coding performance and simple structure, quad-tree based image coding algorithms became one of the most successful approaches for image codecs [8]. For quad-tree based algorithms, the coding length of symbols of significant value is typically close to theoretical entropy, and it also consumes a very small portion of the entire code. In this study, we exploited the relation of different level coefficients in a quad-tree structure, whose magnitudes are usually doubled in higher adjacent wavelet transform level. This feature dominates most kinds of images resulting in stable symbols of significant bit map with searching entropy. Based upon extensive statistic analysis, we present a quad-tree searching model which renders two static codes whose lengths are always close to the entropy. By using this model, we propose a fast and efficient image codec obtaining a 0.2-0.4 db gain over SPIHT for almost all images at various coding rates while also offering significantly reduced computational costs.

The performance of our codec is even comparable with SPIHT with adaptive arithmetic code and JPEG2000. With just a very small loss in performance, our code is far simpler and faster than both SPIHT with adaptive arithmetic code and JPEG2000, which makes our code a very competitive image code alternative for embedded processing communication system where energy and computing resources are limited.

2. PROPOSED QUAD-TREE SEARCHING MODEL

In quad-tree image coding algorithms [9]-[11], the original raw image pixel matrix first undergoes Discrete Wavelet

Transform (DWT) operation, and then quad-trees are formed across wavelet sub-bands. The cost of encoding quad-tree symbols can be divided into two components:

$$Total\ cost = symbols\ of\ significant\ map + symbols\ of\ significant\ values$$

As a matter of fact in the existing image coding methods, total symbols of significant map cost many more bits than symbols of significant values. They usually amount to more than 90% of the total cost in bits. Therefore, how to efficiently encode the significant map is very important. In the existing methods, quad-trees are searched from symbols of significant map resulting from the last coding loop or coding plane. Take degree-2 zero tree [12] as an example, there are two types of symbols of significant map.

- $D(i,j)$: *set of coordinates of all descendants of node* (i,j)
- $L(i,j)$: *set of coordinates of all descendents of node* (i,j) *except offsprings*

In set $D(i,j)$, the occurrence of significant values in its four direct descendants is encoded if it has a significant value. Similarly, in set $L(i,j)$, existence of significant branches is encoded.

The entropy of both cases can be formulated as follows,

$$H = \sum_{i=1}^{16} \rho_i \log(\frac{1}{\rho_i}) \quad (1)$$

where ρ_i indicates probability of case i. To better interpret this entropy, 16 cases can be grouped into 5 possibilities.

P0: No element is significant
P1: Only 1 element is significant
P2: 2 elements are significant
P3: 3 elements are significant
P4: All elements are significant

Therefore, the entropy can be rewritten as the following simple expression.

$$H = -\sum_{i=0}^{4} Pi \log Pi \quad (2)$$

In quad-tree structure, the magnitudes of upper level coefficients are typically twice higher than the adjacent lower level coefficients. This fact will lead to different entropies for searching significant values in set $D(i,j)$ and set $L(i,j)$.

In set $D(i,j)$, the four coefficients in adjacent lower level will be searched if this set has a significant value. According to our extensive tests and analysis, the entropy of the existence of four offspring in set $D(i,j)$ is almost constant for all kinds of natural images at all different coding rates. This is due to the stability of double magnitude relation among adjacent wavelet plateaus in quad-tree structure. According to our tests, the probability distribution P0, P1, P2, P3 for $D(i,j)$ are usually close to 0.0816, 0.5983, 0.2625, 0.0427, and 0.0148.

If we assume that the sub-cases in each group are uniformly distributed, then the corresponding coding entropy is

$$H_D = -\sum_{i=0}^{4} Pi \log \frac{Pi}{C_4^i} = 3.5382 \quad (3)$$

Its corresponding optimal codes for this distribution are 000, 001, 010, 011, 100, 1010, 1011, 1100, 1101, 11100, 11101, 111100, 111101, 111110, 1111110, 1111111. It is worth noting that this code is almost optimal for every image at arbitrary coding rate due to the magnitude relation of adjacent levels in wavelet quad-tree structure. Let us denote these codes as code 1.

In set $L(i,j)$, the four descendant branches excluding the offsprings are searched. The magnitudes of coefficients are typically four times lower than the parent coefficients. So it is high likely that there is no significant branches. According to our tests, the entropy for set $D(i,j)$ is even more stable than the entropy for set $L(i,j)$. The possibility distribution P0, P1, P2, P3, P4 of $L(i,j)$ are around the values of 0.5949, 0.2201, 0.1321, 0.0366,0.0164, respectively.

The corresponding coding entropy based on this probability distribution is

$$H_L = -\sum_{i=0}^{4} Pi \log \frac{Pi}{C_4^i} = 2.4716 \quad (4)$$

The corresponding optimal codes for searching four descendant branches of set $L(i,j)$ are 0, 1000, 1001, 1010, 1011, 11000, 11001, 11010, 11011, 11100, 111010, 111011, 111100, 1111101, 1111110, 1111111. Let us denote these codes as code 2.

Code 1 and Code 2 presented in this paper are almost optimal for every image at arbitrary coding rates. The test results in Fig.1demonstrates that fact.

These three representative images, which are mostly used by image compression tests, are selected here to show that our two static codes are very stable and close to being optimal solutions. As these figures show, the entropy of searching $D(i,j)$ and $L(i,j)$ are nearly constant with a small fluctuation at different code rates, our two codes were very closely following the optimal coding length, especially for code 2 which can get optimal length for almost all code rates in almost all images.

3. THE COMPARISION OF SYMBOL SEARCHING CODE BETWEEN SPIHT AND OUR MODEL

An efficient significant map coder is crucial to the efficiency of the entire image codec. In each bit-plane, a new significant map is generated by symbol searches. In this part, our code is compared with SPIHT, one of the most efficient

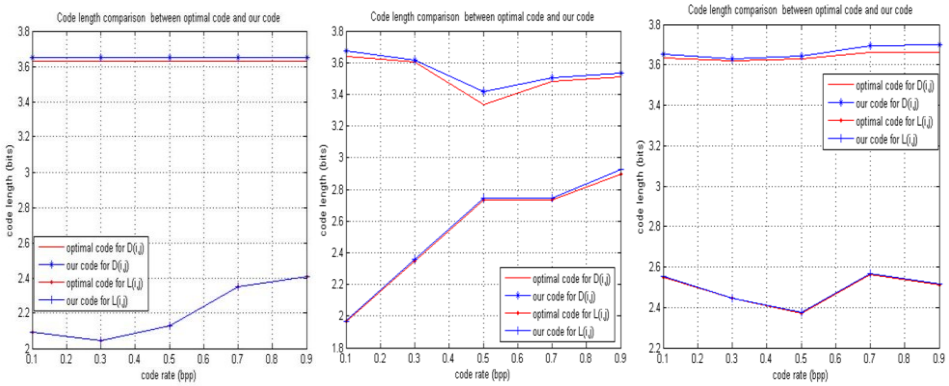

Fig.1. Code Length Comparison Between Optimal Code and Our Code in different images: Lena, Elaine, Living room (from left to right)

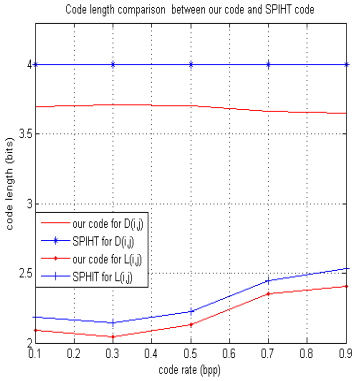

Fig. 2. Code Length Comparison in Lena Image

codecs available, to better understand our code's advantages.

In SPIHT, the length of searching code for $D(i,j)$ is constantly four, which is higher than the actual searching entropy (about 3.5) according to our tests. Our code is designed to be more efficient in representing the quad-tree searching entropy, rendering a code length very close to the actual entropy. Similarly, our code gives shorter length outputs than SPIHT for $L(i,j)$.

Our model takes advantage of the magnitude relationship of adjacent wavelet coefficients. For natural images at various coding rates only a few significant elements usually appear among lower levels in symbol map searching. This fact leads to stable entropy upper-bounds for symbol map searching. The efficiency provided by our model results from our code length, which is closer to this upper-bound than SPIHT. To better understand this improvement, Lena image, which is cited by most researchers, is taken as an example with its average length comparison demonstrated in Fig.2. The code lengths for $D(i,j)$ and $L(i,j)$ are reduced by about 0.4 and 0.125 respectively with a wide variety of coding rates.

More importantly, our proposed model leads to a fast and low-complexity computing process because the image codec can be completed without the need for post-searching entropy coding algorithms such as arithmetic coding. Thus, the proposed method is more favorable for embedded computing and communications platforms.

4. CODING ALGORITHM

In this section, we propose our coding algorithm. First, we define some terminologies, some of which are inherited from SPIHT.

- LSP: List of Significant Pixel
- LIP: List of Insignificant Pixel
- LSB: List of symbols of Significant Bitmap

LSB consists of *D*-symbol and *L*-symbols. The coding algorithm is summarized in the following steps:

1. **Initialization. Initialize the LSP, LIP and LSB. Set Threshold to** 2^n.
2. **Search significant value for** 2^n **in LIP**.
3. **Search significant value for** 2^n **in LSB by using code 1 and code 2**.

For D-symbol the existence of significant value in its four descendants will be coded by code 1, those who are significant are added to LSP, those who are insignificant are added to LIP. The existence of significant value in its four descendant branches followed by these four descendants will be coded by code 2. Then four new D-symbols or one L-symbols are generated, which depend on whether there exists a significant value in the four descendant branches. If L-symbol is generated, it will be added to LSB. If four new D-symbols are generated, those who have significant value will be repeated coded, those who do not have significant value are to be added to LSB.

For L-symbol, the existence of significant coefficients in the four branches is encoded by code 2. The branches that do not have significant value are added to LSB which will be searched by using a successively lower threshold; the branches that have significant value are searched as a new D-symbol.

4. **Code most significant bits in LSP except newly generated significant pixels from previous step**.
5. **Decrease n by 1, go to step 2**.

Our algorithm is similar in style and simplicity to SPIHT, but is more efficient than SPIHT since it adopts two optimal searching codes proposed in Section 2.

5. RESULTS

Extensive tests and analysis were conducted to verify our code's efficiency among a large image set. We used 5-level pyramids constructed with 9/7 tap bi-orthogonal filters. The performance comparison results are shown in Table I.

With low complexities, the performance result shows our code generally outperforms the raw SPIHT by about 0.2 to 0.4dB for all different coding rates. The major reason for this is outcome that our searching code model is more efficient in encoding significant bit map than SPIHT.

TABLE I. QUALITY COMPARISON BETWEEN OUR CODE AND SPIHT

LENA 512X512

Coding Rate	0.1 bpp	0.3 bpp	0.5 bpp	0.7 bpp	0.9 bpp
SPIHT	29.80	34.46	36.83	38.24	39.54
Our Code	29.99	34.75	37.02	38.47	39.75

MANDRILL512X512

Coding Rate	0.1 bpp	0.3 bpp	0.5 bpp	0.7 bpp	0.9 bpp
SPIHT	23.32	27.12	30.12	32.64	35.35
Our Code	23.51	27.38	30.34	33.17	35.59

Furthermore, the performance of our code without adaptive arithmetic code (AAC) is even comparable to SPIHT with adaptive arithmetic coding (SPIHTAAC) and JPEG2000. However, our code can be much faster than SPIHTAAC and JPEG2000, due to the absence of AAC coding in our code. Tables II and III show execution times and performance, respectively.

The results were generated on a PC with an AMD Turion 64x2 TL-60 2.00GHz processor and 2.00GB RAM running the Microsoft Windows Vista operating system. The software implementations for coding images are: JasPer version 1.900.1 source distribution on JPEG-2000 ISO/IEC 15444-1, SPIHT codec C++ version 8.01.

None reversible 9/7 bi-orthogonal wavelet transformation is used for all cases in order to achieve best performance. As these results show, our code has minimal computation overhead, saving about 40% computation time to SPIHTAAC and saving more than 200% computation time when compared to JPEG2000. At the same time, our code experiences only very small performance degradation (averagely about 0.1db in Lena case), since our code does not add adaptive arithmetic coding.

6. Conclusion

In this paper, we propose a new low-complexity and high-speed image coding and decoding method which is especially desirable in embedded computing and communications systems. Our major contribution is a new simple but effective quad-tree searching model for image coding which reduces the computational costs significantly while the coding performance is similar or better when compared to existing main-stream image coding algorithms. Our simulation results demonstrate that the proposed codec surpasses SPIHT by 0.2-0.4db at wide band coding rate. Furthermore, our code's performance is even comparable with SPIHTAAC and JPEG2000 but with much simpler structure and faster speed. This is compelling for embedded communication system where time and energy are more critical. Also our code can be extended to any SPIHT based applications to render a better performance.

TABLE II. RUN TIME COMPARISON (LENA512X512) SPIHT WITH AAC

Coding Rate	0.3 bpp	0.5 bpp	0.7 bpp	0.9 bpp
Compression	**15ms**	**24ms**	**28ms**	**35ms**
Un-compression	**14ms**	**21ms**	**31ms**	**37ms**

OUR CODE WITHOUT AAC

Coding Rate	0.3 bpp	0.5 bpp	0.7 bpp	0.9 bpp
Compression	**7ms**	**12ms**	**15ms**	**17ms**
Un-compression	**4ms**	**13ms**	**13ms**	**16ms**

JPEG2000

Coding Rate	0.3 bpp	0.5 bpp	0.7 bpp	0.9 bpp
Compression	**71ms**	**74ms**	**72ms**	**73ms**
Un-compression	**65ms**	**70ms**	**72ms**	**70ms**

TABLE III. QUALITY COMPARISON WITH LENA512X512

Coding Rate	SPIHT	Our Code	SPIHT w/ AAC	JPEG2000
0.1bpp	29.80	29.99	29.83	29.95
0.3bpp	34.46	34.75	34.96	34.99
0.5bpp	36.83	37.02	37.21	37.27
0.7bpp	38.24	38.47	38.71	3858
0.9bpp	39.54	39.75	39.80	39.79

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, pp. 243–250, June 1996.

[2] M.D. Adams, R.K Ward," JasPer: a portable flexible open-source software tool kit for image coding/processing", *in Proc.IEEE ICASSP '04,*Vol 5, pp.241-244, May 2004

[3] D. Santa-Cruz, et. al, "JPEG 2000 still image coding versus other standards," *in Proc. 45th SPIE Conf. Applications of Digital Image Processing* , San Diego, CA, Vol. 4115, pp. 446–454, Aug. 2000.

[4] W. A. Pearlman, A. Islam, N. Nagaraj and A. Said," Efficient, Low-Complexity Image With Set-partitioning Embedded Block Coder" *IEEE trans. Circuits Syst. Video Technol.*, vol. 14, pp.1219-1235, Nov. 2004

[5] X. Wang, M.J. R. M,J, Y. Wu,'A Time slicing adaptive OFDM system for mobile multimedia communications", *IEEE Trans. Broadcasting* pp.226-234. May 2010.

[6] Y.-R. Tsai and X. Li, "Kasami code-shift-keying modulation for ultrawideband communication systems," *IEEE Trans. Commun*., vol. 55,no. 6, Jun. 2007.

[7] L. Zhang, M. Hauswirth, L. Shu, Z. Zhou, V. Reynolds, and G. Han, "Multi-priority Multi-path Selection for Video Streaming in Wireless Multimedia Sensor Networks," *Ubiquitous Intelligence and Computing*,pp. 439-452,2009

[8] A. Munteanu, J. Cornelis, G. Van de Auwera, and P. Cristea, "Wavelet image compression—The quadtree coding approach," *IEEE Trans.Inform.Technol. Biomed.*, vol. 3, pp. 176–185, Sept. 1999.

[9] Y. Cho and W. A. Pearlman, "Quantifying the Coding Performance of Zerotrees of Wavelet Coefficients: Degree-k Zerotree", *IEEE trans. Signal Processing*, vol. 55, pp. 2435-2431, June 2007.

[10] Y. Cho and W. A. Pearlman, "Quantifying the coding power of zerotrees of wavelet coefficients: A degree-k zerotree model," *in Proc.IEEE ICIP '05*, Sep. 2005.

[11] J.X. Wang, F. Zhang, "Study of the image compression based on SPIHT algorithm", *IEEE Int. Conf. Intelligent Computing and Cognitive Informatics*, pp.130-133 June 2010.

[12] V. N. Ramaswamy, N. Ranganathan, and K. R. Namuduri, "Peformance analysis of wavelets in embedded zerotree-based lossless image coding schemes," *IEEE Trans. Signal Process.,* vol. 47, no. 3, pp.884–889, Mar. 1999.