Adaptive Runlength Coding

Chengjie Tu, *Student Member, IEEE*, Jie Liang, *Student Member, IEEE*, and Trac D. Tran, *Member, IEEE*

***Abstract*—Runlength coding is the standard coding technique for block transform-based image/video compression. A block of quantized transform coefficients is first represented as a sequence of RUN/LEVEL pairs that are then entropy coded—RUN being the number of consecutive zeros and LEVEL being the value of the following nonzero coefficient. We point out in this letter the inefficiency of conventional runlength coding and introduce a novel adaptive runlength (ARL) coding scheme that encodes RUN and LEVEL separately using adaptive binary arithmetic coding and simple context modeling. We aim to maximize compression efficiency by adaptively exploiting the characteristics of block transform coefficients and the dependency between RUN and LEVEL. Coding results show that with the same level of complexity, the proposed ARL coding algorithm outperforms the conventional runlength coding scheme by a large margin in the rate–distortion sense.**

***Index Terms*—Adaptive entropy coding, context modeling, image compression, runlength coding.**

I. Introduction

THE BLOCK-BASED coding scheme of runlength coding coupled with the discrete cosine transform (DCT) is the basis of many international multimedia compression standards from JPEG [1] for still images to the MPEG and H.26X family for video sequences. The beauty of this coding scheme is its simplicity, low computational complexity, low memory requirement, and flexibility on a block-by-block basis. Although current state-of-the-art image coders are wavelet based, block-based coding is still the dominant force in video coding. Besides, block-based coding is preferred in real-time or resource-constrained applications.

Runlength coding represents a quantized DCT block as a RUN/LEVEL sequence that is then coded by various entropy coding techniques. Conventional runlength coding, which is used by JPEG and the MPEG family, codes a (RUN/LEVEL) pair jointly by static Huffman coding or arithmetic coding (AC). The inefficiency of this coding scheme in image coding has been well documented. The DCT-based embedded zerotree (EZ) coder [2] rearranges the DCT coefficients into the wavelet zerotree structure and codes them using set partitioning in hierarchical trees [3]. The **C**ontext-based **E**ntropy coding for **B**lock transform coefficients (CEB) coder [4] uses advanced context-based entropy coding to encode DCT coefficients. Both coders are still constrained in the block-based framework, yet consistently outperform baseline JPEG by a large margin. If pre- and postfiltering are added to the block-DCT-based framework as illustrated in Fig. 1, EZ [5] and CEB [4] achieve competitive rate–distortion (R–D) performances compared with the wavelet-based JPEG2000 coder [6]. Furthermore, annoying block artifacts at low bitrates are suppressed. The cost for implementing pre- and postfiltering is low, since various fast algorithms exist [7].

The inefficiency of conventional runlength coding is mainly a result of too many symbols being involved in coding the (RUN/LEVEL) pairs jointly. Another disadvantage is that it is not adaptive to input statistics, bitrates, and known information (contexts). An obvious solution to the problem is to encode RUN and LEVEL separately instead of jointly by context-based adaptive entropy coding. H26L [8]—the new ITU-T recommendation for video coding at low bitrates—encodes RUN and LEVEL separately for 4×4 DCT blocks by either Huffman coding or AC. However, adaptive context modeling is still absent, and known information has been mostly ignored.

This letter presents an ARL coding algorithm that deals with a small number of symbols by encoding RUN and LEVEL separately. It is highly adaptive, since binary adaptive alternating current (ac) is used for entropy coding. Furthermore, different adaptive models are employed for different contexts to fully exploit the correlation between a symbol and the known information, resulting in more accurate symbol prediction and thus leading to better compression. Detailed context modeling based on the nature of RUN/LEVEL sequences is also presented.

II. Run/Level Representation

The generic approach of runlength coding orders a block of quantized coefficients into a zigzag sequence in the order of increasing frequency. A RUN/LEVEL sequence is generated as follows:

$$(\text{DC})(\text{RUN/LEVEL})\ldots(\text{RUN/LEVEL})(\text{EOB}) \qquad (1)$$

which is then entropy-coded. DC is the value of the direct current (dc) coefficient, which is usually treated seperately. RUN represents the number of consecutive zero ac coefficients prior to a nonzero one, whereas LEVEL is the value of the nonzero ac coefficient. The zigzag order of LEVEL is defined as the index of its corresponding coefficient in the zigzag sequence. The end-of-block (EOB) symbol indicates that there are no additional nonzero ac coefficients in the sequence. We treat EOB as a special RUN symbol.

The most significant property of a DCT block is that the coefficient magnitude typically decreases as the frequency increases. After quantization, only a few coefficients are nonzero, and most of them cluster in the top-left corner of the block.

Manuscript received January 2, 2002; revised July 3, 2002. This work was supported by the National Science Foundation under Grant CCR-0093262. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Ricardo L. de Queiroz.

The authors are with the Department of Electrical and Computer Engineering, The Johns Hopkins University, Baltimore, MD 21218 USA (e-mail: cjtu@jhu.edu; jieliang@jhu.edu; trac@jhu.edu).

Digital Object Identifier 10.1109/LSP.2002.807873

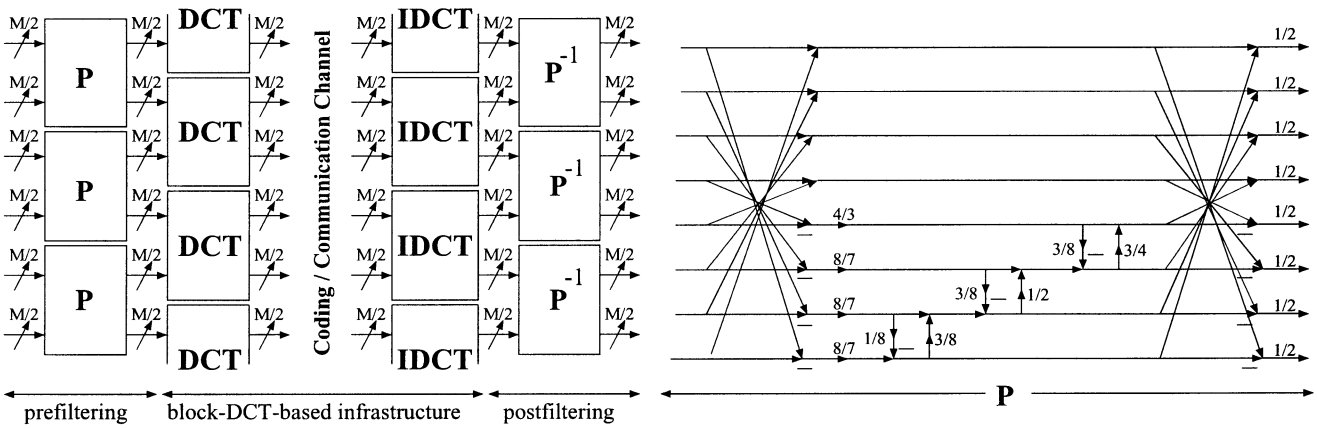

Fig. 1. Pre- and postfiltering. Global framework with block size M (left) and a fast eight-point prefilter $\mathbf{P}$ (right).

By the virtue of zigzag scanning, its RUN/LEVEL sequence is usually short, and EOB represents a large number of zero ac coefficients.

Other properties of RUN/LEVEL sequences include the following.

- Smaller RUN occurs more frequently.
- LEVEL with a smaller magnitude occurs more often.
- The magnitude of LEVEL is expected to be small if its zigzag order is large.
- RUN is more likely to be large if the zigzag order of its preceding LEVEL is large.
- The probability that large RUN followed by LEVEL with a large magnitude is small.
- LEVEL with a large magnitude is usually followed by small RUN.
- EOB is generally preceded by LEVEL with magnitude 1.

All of these properties can be taken advantage of in the context modeling of RUN and LEVEL.

Conventional runlength coding treats a (RUN/LEVEL) pair as one symbol (possibly followed by one or several accessory symbols), and then the symbol is coded by Huffman coding with fixed Huffman tables or by arithmetic coding generally with predefined statistics. Its coding efficiency at least suffers from the following shortcomings.

- Entropy coding involves too many symbols, since the number of possible (RUN/LEVEL) pairs is large. Hence, the resulting entropy codes for most symbols are long.
- The aforementioned properties of RUN/LEVEL sequences are not fully exploited, and most of the known information is ignored.
- Neither Huffman coding with fixed Huffman tables nor arithmetic coding with predefined statistics is adaptive to input images and bitrates, although the statistics of (RUN/LEVEL) pairs are highly image and bitrate dependent. For example, baseline JPEG uses four bits (1010) to encode EOB, which is highly ineffective at low bitrates, since most ac coefficients in this case are quantized to zero.

In our proposed ARL coding scheme, coding efficiency is improved by treating RUN and LEVEL separately and using context-based adaptive AC. For 8×8 blocks, since there are only 63 possible values for RUN and $2m$ possible values for LEVEL, if the maximum magnitude of LEVEL is m, then the number of symbols is small. It is also much easier to model RUN and LEVEL separately than jointly, allowing better entropy coding. All aforementioned properties of RUN/LEVEL sequences are taken into account by using different adaptive models for different contexts. Adaptive AC optimizes the bit budget for each symbol automatically for different images and different bitrates. Proper context modeling as described in Section III turns out to be the key to the success of ARL coding.

TABLE I
BINARIZATION OF RUN AND LEVEL SYMBOLS

RUN	Binarization	LEVEL	Binarization
EOB	1	1	0 1
0	0 1	-1	1 1
1	0 0 1	2	0 0 1
2	0 0 0 1	-2	1 0 1
3	0 0 0 0 1	3	0 0 0 1
4	0 0 0 0 0 1	-3	1 0 0 1
⋮	⋮	⋮	⋮

III. CONTEXT MODELING

A. Binarization of RUN and LEVEL

The entropy coding engine for ARL coding is binary adaptive AC, the simplest and fastest version of adaptive AC. It approaches the underlying statistics very promptly and thus suffers little from context dilution.

To encode a nonbinary symbol by binary AC, the symbol must be binarized first. Binary AC is then employed to each binary symbol, or bin for short. Simple binarization rules for RUN and LEVEL are illustrated in Table I: EOB is binarized as "1"; other RUN is binarized as $(\text{RUN} + 1)$ "0's" followed by a "1"; LEVEL is binarized as "0" if it is positive or "1" if it is negative followed by $(|\text{LEVEL}| - 1)$ "0's" and a "1", where $|\text{LEVEL}|$ is the magnitude of LEVEL. Here, shorter binary sequences are used for smaller RUN symbols and LEVEL symbols with small magnitudes, since they occur more frequently. Since the statistics of different bins may differ greatly, different models are usually used for different bins to maximize coding efficiency.

TABLE II
COMPARISON OF CODING PERFORMANCES IN PEAK SIGNAL-TO-NOISE RATIO (IN DECIBELS)

	8 × 8 DCT				8 × 8 DCT with Pre/Post-Filtering		9/7 wavelet
bpp	Baseline JPEG	AC-JPEG	EZ	ARL	EZ	ARL	JPEG2000-SL
Lena (512 × 512)							
0.125	27.6	28.4	29.6	29.7	31.0	31.0	**31.2**
0.25	31.6	31.9	32.8	32.8	34.1	34.1	**34.3**
0.5	34.9	35.1	36.2	36.2	37.1	37.0	**37.4**
1.0	37.9	38.2	39.6	39.6	39.9	40.0	**40.6**
Goldhill (512 × 512)							
0.125	26.7	27.2	27.9	27.8	**28.6**	**28.6**	**28.6**
0.25	29.1	29.6	30.0	30.0	**30.7**	**30.7**	**30.7**
0.5	31.6	32.1	32.7	32.7	**33.3**	**33.3**	**33.3**
1.0	34.5	35.0	36.2	36.2	36.6	**36.7**	**36.7**
Barbara (512 × 512)							
0.125	23.1	23.6	24.6	24.4	**25.9**	25.4	25.6
0.25	25.2	25.6	27.2	27.1	**29.0**	28.5	28.6
0.5	28.4	28.9	31.1	31.0	**32.9**	32.7	32.5
1.0	33.2	33.7	36.2	36.1	**37.6**	37.5	37.4
Bike (2048 × 2560)							
0.125	24.1	24.5	25.4	25.2	26.1	25.7	**26.5**
0.25	27.2	27.6	28.7	28.3	29.3	28.7	**29.8**
0.5	30.4	30.8	32.7	32.1	33.1	32.4	**33.7**
1.0	34.4	34.3	37.4	36.6	37.4	36.6	**38.3**
Cafe (2048 × 2560)							
0.125	19.7	19.9	20.2	20.2	20.7	20.6	**20.9**
0.25	21.7	22.2	22.7	22.7	23.2	23.1	**23.3**
0.5	24.6	25.0	26.7	26.2	26.8	26.5	**27.0**
1.0	28.7	29.0	31.4	31.1	31.8	31.4	**32.3**
Woman (2048 × 2560)							
0.125	25.4	25.9	26.9	26.9	27.2	27.1	**27.5**
0.25	28.2	28.4	29.5	29.5	30.1	29.8	**30.2**
0.5	30.9	31.3	33.2	33.1	33.6	33.5	**33.8**
1.0	34.7	35.0	37.9	37.7	38.0	37.9	**38.7**
First luminance frame of News (176 × 144)							
0.125	21.0	21.5	23.0	23.4	23.5	**24.1**	23.3
0.25	24.4	24.7	25.8	26.4	26.5	**27.0**	26.5
0.5	27.9	28.4	30.1	30.5	30.6	**30.9**	30.4
1.0	32.5	33.0	35.9	35.9	36.3	36.2	**36.6**
First luminance frame of Glasgow (176 × 144)							
0.125	21.1	21.7	22.8	23.3	23.4	**23.7**	23.0
0.25	23.7	24.1	24.9	25.2	25.5	**25.7**	25.0
0.5	26.4	26.8	27.9	28.0	28.2	**28.5**	27.7
1.0	29.3	29.7	32.3	32.5	**32.6**	**32.6**	32.5

B. Context Models for DC Coefficients

The dc coefficient of a block is predicted as the mean of the reconstructed dc coefficients of its left and top block neighbors. Three models are used depending on whether the quantized residue is zero or not: one for $z = 0$, one for $z = 1$, and one for $z = 2$, where z is the number of nonzero quantized residues of the two neighboring blocks. We denote this as

$$(z = 0)\,(z = 1)\,(z = 2). \tag{2}$$

The idea behind this is that if the dc coefficients of the neighboring blocks can be predicted well, the probability of an accurate estimation of the current dc coefficient is also high. If the quantized dc residue is nonzero, it is coded in the same manner as that of a regular LEVEL symbol.

C. Context Models for RUN

Define $f = 0, 1,$ or 2 if neither of, either of, or both the left and top block neighbors have nonzero quantized ac coefficients. Here, f estimates how flat the current block is. If $f = 0$, the first RUN of the current block is most likely to be EOB, which means the block is flat (every ac coefficient is quantized to zero). The context modeling for coding the first bin of the first RUN is

$$(f = 0)\,(f = 1)\,(f = 2). \tag{3}$$

Two more models are needed for coding the first RUN: one for the second bin and one for all the remaining bins.

Other RUN symbols are coded conditioned on the magnitude and the zigzag order of the preceding LEVEL, denoted as m and l, respectively. Based on the aforementioned observation that RUN is more likely to be small if m is large or l is small, the context modeling is chosen as

$$\begin{aligned}&(l < 6 \text{ and } m = 1)\,(l < 6 \text{ and } m > 1)\\ &(6 \le l < 15 \text{ and } m = 1)\,(6 \le l < 15 \text{ and } m > 1)\\ &(l \ge 15).\end{aligned} \tag{4}$$

Different models are used for the first bin, the second bin, and all remaining bins. Including the five models for the first RUN, a total of 20 models are used for RUN symbols.

D. Context Models for LEVEL

The first bin of LEVEL contains its sign information: one single model is employed here. The remaining bins are coded

conditioned on l, the zigzag order of LEVEL, and r, the current RUN. The detailed context modeling is

$$(l=0)(0<l<3)(3\le l<15 \text{ and } r<3)\\ (15\le l \text{ or } 3\le r). \quad (5)$$

Different models are used for the second bin and all the following bins. The modeling reflects the fact that LEVEL with large zigzag order usually has a small magnitude and large RUN is generally followed by LEVEL with a small magnitude. Including the single model for the first bin, altogether we need nine models to encode LEVEL symbols.

Comparing to conventional runlength coding, the proposed ARL coding scheme does not need more memory, since only 32 binary models are involved, and there is no need to buffer any Huffman tables or predefined statistics.

IV. Coding Results

An image coder is implemented to verify the R–D performance of ARL coding. The coder is similar to baseline JPEG: an image is mapped to nonoverlapped 8×8 coefficient blocks; each block is transformed, uniformly quantized, and coded by runlength coding. Besides the slight difference in dc prediction as discussed in Section III-B, the only difference is that ARL coding replaces JPEG's runlength coding. The advanced version has pre- and postfiltering between neighboring blocks as shown in Fig. 1 turned on to improve coding efficiency and to eliminate blocking artifacts.

Three groups of eight-bit grayscale images are used: 512×512 Lena, Goldhill, and Barbara; 2048×2560 Bike, Cafe, and Woman; and the first luminance frames of quarter common intermediate format (176×144) sequences News and Glasgow. Baseline JPEG (with optimized Huffman coding) and JPEG with arithmetic coding (AC-JPEG) [1], the improved EZ coder [5], and JPEG2000 in the single-layer mode (JPEG2000-SL) [6] serve as benchmarks. As many levels of dyadic wavelet decompositions as possible are employed in JPEG2000-SL and for EZ's dc subbands. The popular biorthogonal 9/7 filters are used.

Table II tabulates the coding results. With the same level of complexity, our proposed ARL coder consistently outperforms baseline JPEG (0.5–3.4 dB and 1.8 dB on average) and AC-JPEG (0.2–2.9 dB and 1.4 dB on average) by a large margin for all images at all bitrates. This confirms that ARL coding is much more efficient than conventional runlength coding. If pre- and postfiltering are added, ARL is 0.9–4.3 dB (2.4 dB on average) better than baseline JPEG and 0.7–3.8 dB (2 dB on average) better than AC-JPEG. Most of the time the ARL coder is comparable to EZ and even to JPEG2000-SL despite the fact that EZ and JPEG2000 are a lot more complex. Note that the ARL coder is penalized significantly by its simple dc treatment (only simple dc prediction is used, since it is not our main focus in this letter), whereas both EZ and JPEG2000-SL employ many levels of wavelet decomposition to the lowpass (dc) subbands. This explains ARL's inferior R–D performances for high-resolution images such as Bike. For the low-resolution 176×144 images where there is not much correlation left in the dc subbands, ARL outperforms other coders.

V. Conclusion

We have presented a novel ARL coding algorithm. Benefitting from advanced context modeling and adaptive arithmetic coding, the proposed coding scheme demonstrates much improved R–D performance comparing to the conventional runlength coding scheme while maintaining the same level of computational complexity.

References

[1] W. B. Pennebaker and J. L. Mitchell, *JPEG Still Image Data Compression*. New York: Van Norstrand Reinhold, 1992.

[2] Z. Xiong, O. Guleryuz, and M. T. Orchard, "A DCT-based embedded image coder," *IEEE Signal Processing Lett.*, vol. 3, pp. 289–290, Nov. 1996.

[3] A. Said and W. A. Pearlman, "New, fast, and efficient image codec based on set partitioning in hierarchical trees," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 6, pp. 243–249, June 1996.

[4] C. Tu and T. D. Tran, "Context based entropy coding of block transform coefficients for image compression," in *Proc. SPIE Applications of Digital Image Processing XXIV*, San Diego, CA, Aug. 2001, pp. 377–389.

[5] T. D.Tran and T. Q. Nguyen, "A progressive transmission image coder using linear phase uniform filterbanks as block transforms," *IEEE Trans. Image Processing*, vol. 8, pp. 1493–1507, Nov. 1999.

[6] "JPEG-2000 VM3,1A software," ISO, ISO/IEC JTC1/SC29/WG1 N1142, Jan. 1999.

[7] J. Liang, C. Tu, and T. D. Tran, "Fast lapped transforms via time-domain pre- and post-filtering," in *Proc. ICICS*, Singapore, Oct. 2001.

[8] H26L Test model long term number 8 (TML-8) draft0, ITU-T Study Group 16 (VCEG), June 2001.