

A. Class of Reversible Variable Length Codes for Robust Image and Video Coding

Jiangtao Wen and John D. Villasenor

Electrical Engineering Department
University of California, Los Angeles
gwen, villa@icsl.ucla.edu

Abstract

We describe a class of parameterized reversible variable length codes that have length distributions identical to Golomb-Rice codes and exp-Golomb codes. The pdfs to which these codes correspond are well matched to statistics of image and video data, thus enabling an increase in robustness to channel errors with no penalty in coding efficiency. These codes are applicable to MPEG-4 and other algorithms that aim to use variable length codes in error-prone environments.

1 Introduction

The overwhelming majority of current image and video coding standards and systems use some form of run length and entropy coding to obtain efficient compression. This results in variable length codes that are very vulnerable to transmission over a noisy channel because of the synchronization losses that can accompany bit errors. Although in principle one could increase robustness by replacing variable length codes with fixed length codewords such as those used in some types of vector quantization, fixed length codes are not typically as efficient as traditional entropy codes in complexity- and memory-constrained environments. In addition, in developing next-generation solutions for robust image and video communications there is strong incentive in some of the image and video coding standards organizations to preserve the basic run-length/Huffman coding framework used in algorithms such as MPEG-1, MPEG-2, H.263, and MPEG-4.

The above considerations have led to a growing level of interest in reversible variable length codes (RVLCs). The idea behind RVLCs is that decoding can be performed by processing the received bit-stream either forwards or backwards. For example, a decoder can begin by processing the received bit-stream in the forward direction, and upon detecting an error, proceed immediately to the end of the bit-stream and decode in the reverse direction. This, and similar strategies can be used to significantly reduce the effects of bit errors on the fidelity of the decoded video. Although entropy coding has received extensive attention for many decades, RVLCs have received

significant attention only recently, particularly in association with work in the MPEG-4 community.

One of the first publications describing RVLCs in detail was by Takishima, Wada and Murakami in [1]. Takishi *et al.* studied the conditions for the existence of RVLCs and proposed algorithms for their construction. Most recently, Toshiba and Ericsson [6] proposed two different schemes for constructing RVLCs with systematic structures that enable easy coding and decoding. The principal contribution of the present paper is the introduction of RVLCs that have identical code length distributions to previously known, non-reversible VLCs that are known to be near-optimal for pdfs that occur in coding of image data. The RVLCs presented here are parameterized to allow them to be adapted to match a wide range of pdfs, and enable the advantages of two-way decoding while retaining the efficiency of traditional (non-reversible) variable length codes.

2 Code Description

We begin by presenting a reversible code with the same length distribution as Golomb-Rice codes. The ideas behind Golomb-Rice codes were originally proposed in [2] and [3], and have recently been applied for coding of prediction errors in lossless image coding applications [4]. Golomb-Rice codes are nearly optimal for coding of exponentially distributed non-negative integers, and describe an integer n in terms of a quotient and a remainder. For simplicity, the divisor is often chosen to be a power of 2, 2^k , and is parameterized by k . The quotient can be arbitrarily large and is expressed using a unary representation; the remainder is bounded by the range $[0, 2^k - 1]$ and is expressed in binary form using k bits. For example, for a Golomb-Rice code with $k = 2$ the number 9 could be represented as 110 01. The "prefix" of the codeword, 110, identifies the quotient of $9/2^2$ as having value 2. The "suffix", 01 is a 2-bit binary expression of the remainder. Table 1a gives Golomb-Rice codes for the first several integers for two choices of the parameter k .

To obtain an equivalent length reversible code, one can simply replace the prefix of each Golomb-Rice

codeword with a prefix that begins and ends with a “1”, with all other bits equal to “0”. The exception to this is the prefix of length one, which is set to “0”. The suffix in the RVLC remains the same as the suffix in the corresponding Golomb-Rice code. RVLCs constructed according to these rules are shown in Table 1a for $k = 1$ and $k = 2$, and it is clear from the table that the length distributions of the RVLCs and the corresponding Golomb-Rice codes are identical. Although it is only the prefix, as opposed to the entire codeword, that is symmetric, these codes can easily be decoded bidirectionally because the non-symmetric portions of the codewords have fixed length. Figures 1a and 1b provide a comparison of the code trees corresponding to the $k = 1$ Golomb-Rice code and the corresponding RVLC from Table 1a.

In contrast with the Golomb-Rice code in which the number of codewords at each length is constant, it is also possible to construct codes in which the number of codewords of a given length grows exponentially with length. Compression of run lengths using such codes was described in a paper by Teuhola [5] using the term “exponential-Golomb” coding. Exp-Golomb codes are matched to pdfs having a higher peak and wider tails than typical exponential pdfs. Such a pdf is very well matched to the run-length coded data that occur in quantized image transforms. Exp-Golomb codes can be parameterized according to k , the number of bits in the suffix of the codeword. Table 1b illustrates the exp-Golomb code for $k = 1$. It is possible, though less straightforward, to construct a reversible code that has the same length distribution as an exp-Golomb code. To do this we again impose the constraint that the first and last bits of the prefix be “1”. As before, the prefix of length one is set to “0”. We require that all odd-indexed bits in the prefix, with the exception of the first and last bit, be “0”. For example, in all prefixes of length 5, the third bit is “0”, and the first and fifth bits are “1”. The even-indexed bits are allowed to vary arbitrarily, allowing $2^{(l-1)/2}$ possible prefixes of length l , where l is odd. In constructing the code, each prefix is concatenated with the 2^k distinct suffixes of length k . Table 1b gives an RVLC constructed according to these rules. Again, it is clear that the length distribution of the RVLC is identical to that of the corresponding reversible code. Code trees for the $k = 0$ exp-Golomb code and corresponding reversible code are shown in Figures 1c and 1d.

3 Error Handling

Since codewords in both the reversible exp-Golomb and reversible Golomb-Rice codes have fixed length suffixes of size k , any bit errors occurring in these suffixes will not result in error propagation. Errors in the prefixes, however, can result in error propagation or equivalently, loss of synchronization. For the reversible Golomb-Rice codes, any error at any position within the prefix will cause loss of synchronization, while for the reversible exp-Golomb codes, error propagation will occur if and only if bit errors occur in odd-indexed positions of the prefix. Any single bit er-

ror can be classified as either a “propagating bit error” or a “non-propagating bit error” depending on where it occurs.

As an example of how these issues effect the exp-Golomb codes, consider a source of non-negative integer symbols i , $i \geq 0$, each with probability $p(i)$. The average length (in bits/symbol) after coding this source using the reversible exp-Golomb code will be $L = \sum_{i=0}^{\infty} p(i)l(i)$, where $l(i) = k+1+2\lceil \log_2(1+i/2^k) \rceil$ is the length of the reversible exp-Golomb code with parameter k for integer i . If a single bit error is applied to the bitstream produced at the coder output, the probability that this error will occur in a codeword representing integer i is $p(i)l(i)/L$. Similarly, the probabilities of the error occurring in either the odd or even positions in the prefix of codeword i are $[p(i)l_o(i)]/L$ and $[p(i)l_e(i)]/L$ respectively, where $l_o(i)$ and $l_e(i)$ are the number of even and odd positioned bits in the prefix of codeword i , so $l_e(i) + l_o(i) + k = l(i)$.

In the case where the source pdf satisfies $p(i) = 2^{-l(i)}$ and is therefore matched to the reversible exp-Golomb code, the probability that a single bit error will be non-propagating is $\sum_{i=0}^{\infty} (k+i)2^{-i-1}/L = (k+1)/(k+3)$. It can be shown that this is identical to the probability that a randomly applied single bit error will be non-propagating for a nonreversible exp-Golomb code. Similarly, the probability that an applied bit error will be non-propagating is $k/(k+2)$ for both reversible and non-reversible Golomb-Rice codes. When a propagating error occurs in a nonreversible code, it is well known that the decoder can usually re-establish synchronization, though perhaps with the insertion or deletion of symbols. For reversible codes, however, the decoder can not generally resynchronize when there is only one propagating bit error; a second propagating error is needed to enable resynchronization. Therefore, in general error propagation will be more severe for reversible codes than non-reversible codes with the same code length distribution. This is balanced by the fact that when, due to multiple errors, resynchronization does occur, a reversible code can permit more information to be extracted at the decoder regarding which portions of the decoding result are correct.

There are many possible decoder strategies for exploiting reversible codes to improve robustness, involving the usual tradeoffs between computational complexity and performance. As an example, consider a system in which the number of bits to be decoded is known *a priori*, and in which some of the nodes of the code tree are invalid and can serve as “traps” to detect errors. This occurs in some video coding algorithms that involve transmission of entropy coded motion vectors in a packet of information whose length is given in a header. Out-of-range motion vectors serve as the traps for invalid data. One simple decoding strategy in this case involves having both the forward and backward decoders keep track of a bitstream pointer to determine where to stop decoding. The forward decoder

will stop when the pointer reaches (or, in the case there are errors, proceeds past) the last bit before the end of the packet; the backward decoder will stop when the pointer reaches (or, if errors have occurred, points to a location preceding) the first bit of the packet. The decoders will also save information regarding the location of any invalid data found, and perform operations necessary for successful subsequent error checking when receiving "traps". After forward and backward decoding is complete, if both decoders stop at the correct positions, the two decoded symbol sequences are compared and those portions which are identical are used for subsequent decoding. If only one of the two decoders (decoder A) stops at the correct position, the initial portion (before the first detected error) of the symbol stream it produces will be considered correct. The final portion of the symbol stream can also be compared with the initial output from decoder B, and the consistent symbols can be considered correct. If neither decoder stops at the right position, then conventional forward only decoding can be used, with data after the first detected error discarded or flagged as potentially incorrect.

4 Experimental Results

To explore the error-robustness of the proposed codes, we used the RVLCs in Table 1b for entropy coding of the runs and levels of the 512 by 512 Lena image after performing a block DCT, zigzag scanning and quantization with the standard JPEG quantization tables. The JPEG "Q-factor" was adjusted to give a bit rate after entropy coding of .5 bpp. To simplify the mapping of the codewords in Table 1b, runs and levels were coded separately, though it is also relatively straightforward to apply the entries in Table 1b to joint coding of (run/level) pairs. Coded images were subject to corruption using a binary symmetric channel with bit error rates (BER) of 10^{-4} and 10^{-3} . At each BER, 1000 runs of the simulation were performed. Decoding was performed first in the forward direction only, and subsequently in both the forward and reverse directions. This allows a direct assessment of the benefit of applying bidirectional decoding. At a BER of 10^{-4} , the bidirectional decoding gave an image domain PSNR that was on average 2.2 dB superior to the PSNR obtained using forward decoding only. The average PSNR improvement at 10^{-3} was .9 dB.

We also carried out, in collaboration with Raj Talluri and Yashoda Nag of Texas Instruments, an MPEG-4 core experiment to examine the performance of the reversible exp-Golomb codes for the coding of motion vectors in a video coding system. The syntax of the bitstream, as illustrated for one packet in Figure 2, is provided by the MPEG-4 verification model (VM 7.0) approved in April 1997 in Bristol, UK ([7]). The modifications made to VM 7.0 as part of the experiment are as follows: First, instead of coding COD and MCBPC information separately, in our experiment, they are coded jointly with a symmetrical (and thus reversible) VLC. These VLCs resemble, but are

not identical to those given in Table 1 (the differences arise from the need to match the code length distributions in the VM as closely as possible). The second modification is to replace the VLC in the VM with reversible exp-Golomb codes with parameter $k = 1$ and $k = 2$ from Table 1b. For each k , the exp-Golomb code of parameter $k - 1$ is used to code the amplitude of the motion vector, with an additional bit added at the end of each codeword to convey the sign. The last bit of the codeword for motion vector 0 is assigned to 1 to avoid emulation of the MPEG-4 resynchronization marker by the motion data.

We compared the modified coder with the VM 7.0 in terms of coding efficiency and error robustness for 5 QCIF format video sequences, each under 8 different error conditions including several binary symmetric channels, bursty channels, and packet lossy channels. For each error condition, 50 simulations were performed, each involving the decoding of 100 frames. No forward error correction was used in the simulations. More detailed description of the simulation conditions can be found in [8]. Comparison of the sizes of compressed sequences shows that the overhead introduced by replacing the non-reversible VLC in VM 7.0 with the RVLC is only 1.5% in average, and ranges from -2.2 % to 4.6 %. For sequences such as Silent and Foreman, the reversible exp-Golomb codes for the motion data are actually more efficient than the non-reversible motion vector VLC in VM 7.0. These sequences include rapid motion, generating larger motion vectors which require longer VLCs (which are more efficiently represented using exp-Golomb codes) to represent. The average PSNR improvement after using the RVLC was about 0.3 dB. The improvement tended to be larger for channels of lower quality. A more detailed description of the experimental results is given in [9].

5 Conclusions

We have introduced codes in which the codeword prefixes are variable length and are constructed using symmetry constraints. The codeword suffixes are non-symmetric but have fixed length. This approach enables the design of parameterized RVLCs that have length distributions that are identical to Golomb-Rice and exp-Golomb codes, and that are therefore well matched to the pdfs occurring in image and video data. When applied to image coding in a JPEG-like framework and video coding as part of an MPEG-4 core experiment carried out in collaboration with Texas Instruments, these codes allow significant improvements in PSNR during transmission over noisy channels.

6 Acknowledgements

This work was supported by Texas Instruments and by DARPA/ITO contract DABT63-95-C-0100. We thank Raj Talluri and Yashoda Nag of Texas Instruments for collaborating with us to define and carry out

the MPEG-4 core experiment on RVLCs.

References

- [1] Y. Takishima, M. Wada, H. Murakami, "Reversible Variable Length Codes," *IEEE Trans. Comm.*, vol. 43, No. 2/3/4, pp. 158-162, 1995.
- [2] S.W. Golomb, "Run-length encodings," *IEEE Trans. Inf. Theory*, vol. IT-12, pp. 399-401, July 1966.
- [3] R.F. Rice, "Some practical universal noiseless coding techniques," Tech. Rep. JPL-79-22, Jet Propulsion Laboratory, Pasadena, CA, March 1979.
- [4] M.J. Weinberger, G. Seroussi, and G. Sapiro, "LOCO-I: A low complexity, context-based lossless image compression algorithm," *Proc. 1996 IEEE Data Comp. Conf.*, Snowbird, UT, pp. 140-149, April, 1996.
- [5] J. Teuhola, "A Compression Method for Clustered Bit-Vectors," *Information Processing Letters*, vol. 7, pp. 308-311, October 1978.
- [6] ISO/IEC JTC1/SC29/WG11 N1383, "Description of Error Resilient Core Experiments," Nov. 1996.
- [7] ISO/IEC JTC1/SC29/WG11 N1642, "MPEG-4 Video Verification Model Version 7.0", Bristol, April 1997.
- [8] ISO/IEC JTC1/SC29/WG11 N1646, "Description of error resilient core- experiments," May, 1997.
- [9] ISO/IEC JTC/SC29/WG11 M2382, "Report of results on core experiments on error resilience for motion data with structured RVLC - E8," July, 1997. Also available from <http://www.icsl.ucla.edu/~ipl/page2.html>

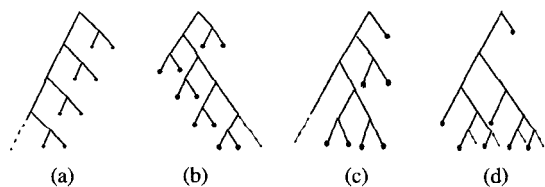


Figure 1: Comparison of codetrees. (a) Golomb-Rice code ($k=1$); (b) Reversible Golomb-Rice code ($k=1$); (c) Exp-Golomb code ($k=0$); (d) Reversible exp-Golomb code ($k=0$).

Resyn. Marker	MB No.	QP	Header Data (COD + MCBPC)	Header Marker	MV Data	Motion Marker	DCT Header	DCT Data	Resyn. Marker
---------------	--------	----	---------------------------	---------------	---------	---------------	------------	----------	---------------

Figure 2: Packet structure for MPEG-4 core experiment. COD - Coded macroblock indication. MCBPC - Macroblock type and coded block pattern for chrominance. MV - Motion vector. QP - Quantization parameter. MB No.- Index number to the first macro block in the packet.

	k=1				k=2			
	Golomb-Rice		RVLC		Golomb-Rice		RVLC	
	Prefix	Suffix	Prefix	Suffix	Prefix	Suffix	Prefix	Suffix
0	0	0	0	0	0	00	0	00
1	0	1	0	1	0	01	0	01
2	10	0	11	0	0	10	0	10
3	10	1	11	1	0	11	0	11
4	110	0	101	0	10	00	11	00
5	110	1	101	1	10	01	11	01
6	1110	0	1001	0	10	10	11	10
7	1110	1	1001	1	10	11	11	11
etc								

Table 1(a)

	Exp-Golomb		RVLC	
	Prefix	Suffix	Prefix	Suffix
0	0	0	0	0
1	0	1	0	1
2	100	0	101	0
3	100	1	101	1
4	101	0	111	0
5	101	1	111	1
6	11000	0	10001	0
7	11000	1	10001	1
8	11001	0	10011	0
9	11001	1	10011	1
10	11010	0	11001	0
11	11010	1	11001	1
etc				

Table 1(b)

Table 1: (a) Golomb-Rice and reversible Golomb-Rice code; (b) Exp-Golomb and reversible codes (odd positioned prefix bits of reversible exp-Golomb codes are shown in bold italic fonts).