Memory Efficient Lossless Compression of Image Sequences with JPEG-LS and Temporal Prediction

Zhe Wang, Debasish Chanda, Sven Simon
Dept. of Parallel Systems, University of Stuttgart
Universitaetsstrasse 38, 70569 Stuttgart, Germany
Email: wangze@ipvs.uni-stuttgart.de

Thomas Richter
RUS Computing Center, University of Stuttgart
Allmandring 30a, 70550 Stuttgart, Germany
Email: richter@rus.uni-stuttgart.de

Abstract— **In this paper, a lossless encoder for image sequences based on JPEG-LS defined for still images with temporal-extended prediction and context modeling is proposed. As embedded systems are one important field of application of the codec, on-line lossy reference frame compression is used to reduce the encoder's memory requirement. Variations of the pixel values in the reference frame due to lossy compression are acceptable since the predictor provides only estimations of the pixel values being encoded in the current frame. Larger variations decrease the final lossless compression performance of the encoder such that a trade-off between the memory requirement and the overall compression ratio is required. Different compression algorithms for the reference frame, including JPEG, JPEG 2000 and near-lossless JPEG-LS, and their impacts on the memory requirement and the overall lossless compression ratio have been studied. Experimental results show 9.6% or more gain in lossless compression ratio compared to applying the standard JPEG-LS frame-by-frame and 80% reduction in the encoder buffer size compared to storing the uncompressed reference frame.**

I. Introduction

Today, the common approaches to lossless video compression are JPEG 2000 [1], [2] e.g. for digital video editing and distribution, and JPEG-LS [3] e.g. for medical image archiving. JPEG-LS in general slightly outperforms JPEG 2000 regarding lossless compression performance [4], [5]. Moreover, the well known low computational complexity of JPEG-LS makes it more suitable for real-time implementation. Motion JPEG 2000 [2] does not employ a temporal prediction, neither does the frame-by-frame compression of JPEG-LS. Although intra-only compression saves memory and allows fast access to a particular frame in the compressed sequence (frame-selectable editing), high temporal correlations exist in adjacent frames especially when the video comprises a static background, which is typical e.g. for surveillance sequences.

In application fields like embedded systems, signal processing algorithms typically suffer from complexity-, memory- and power-constraints. Due to the complexity constraint, operations such as motion estimation are infeasible in certain situations where real-time compression on low complexity hardware is necessary. In addition, limited on-chip memory may prohibit the buffering of even one single reference frame at a high spatial resolution, while off-chip memory results in other critical implementation issues like external high memory bandwidth in the GByte/s domain. Power dissipation is another critical issue especially for embedded image capturing systems e.g. equipped on mobile devices or spacecrafts. The complexity-, memory- and power-constraints together with the fundamental requirement of data volume reduction while retaining image quality pose a considerable challenge on the design of a memory efficient, low complexity and lossless compression algorithm for image sequences.

Authors of [6], [7] addressed the external memory bandwidth and power dissipation issues in video encoders due to reference frame access. In their approach the reference luminance frame is lossy compressed using the min-max scalar quantization [6] before being stored in the external memory. The encoder fetches the lossy compressed macroblocks from the memory for motion estimation in inter-coding mode. The encoder in [7] additionally stores the quantization errors in the external memory and recovers the lossless reference frame for motion compensation to avoid drift. Although reductions of up to 39% in memory bandwidth and 31% in power consumption were achieved [7], storing both the lossy reference frame and the quantization error indicates that the total amount of required memory has not been effectively reduced.

One of the earlier efforts on real-time embedded lossless video compression was reported in [8], where the FELICS algorithm [9] was used to compress 60Hz 1920×1080 video to reduce memory bandwidth in LCD panel chips. FELICS uses hardware-efficient Golomb coding. However this approach did not employ temporal prediction, and as reported in [10], the compression performance of FELICS is 10~20% below that of JPEG-LS. Besides, the context-based sequential selection [9] of Golomb parameter k (fixed at 2 in [8]) creates an issue for memory efficient hardware implementation.

To obtain improved lossless compression efficiency on videos over JPEG-LS, authors of [11] proposed a scheme using motion compensation and a spatial-temporal least mean square error (LMSE) predictor followed by the JPEG-LS entropy coding procedure. A compression gain of about 10% over the standard JPEG-LS was achieved. However apart from the computationally intensive motion estimation, it is well known that the computational complexity of an LMSE predictor is prohibitive for low complexity real-time implementations.

A computationally efficient temporal extension to JPEG-LS was reported in [12]. In this approach, a decision is adaptively made for each pixel based on the spatial and temporal variations in the causal neighborhood of the current

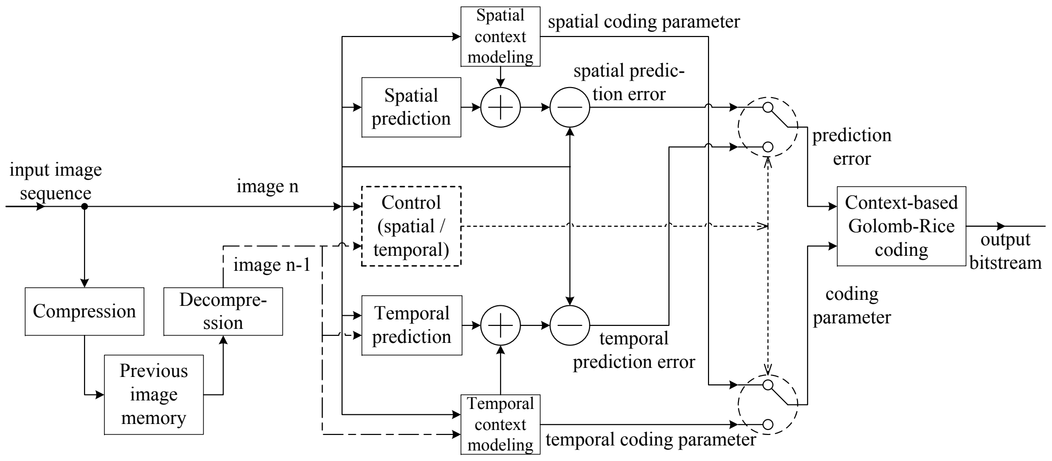

Fig. 1. Block diagram of the proposed lossless encoder for image sequences.

pixel. No side information (e.g. motion vector) is required. Approximately 15% improvement over the standard JPEG-LS on widely used test sequences is achieved at nearly the same computational complexity as JPEG-LS. This approach provides good results on the class of image sequences where the background is static and the foreground object moves slowly. For cost-effective real-time hardware implementations, this approach is a reasonable option. For sequences where objects or backgrounds are moving fast, motion estimation and compensation can be performed as a preprocessing step if enough computational power is allowed. However, the memory efficiency issue still needs to be addressed since this approach requires a reference frame to be stored by the encoder.

In this paper, the trade-off between the required memory for the reference frame and the overall lossless compression gain from temporal prediction is investigated. We propose a memory efficient, low complexity lossless compression algorithm for image sequences based on reference frame compression. The pixel predictor adaptively selects its prediction neighbors from the current frame or the reference frame. The reference frame is compressed and then the compressed bitstream is stored in the on-chip memory to improve memory efficiency of the encoder. The compressed reference frame is then partially decompressed and used in the temporal prediction mode for the current frame. Actual image data of the current frame (spatial / temporal residuals) are encoded without loss.

The paper is organized as follows. Section II presents the proposed lossless compression algorithm based on temporal-extended JPEG-LS and on-line reference frame compression. Section III discusses different options for reference frame compression including JPEG, JPEG 2000 and near-lossless mode of JPEG-LS. Experimental results are presented in Section IV. Finally Section V concludes this paper.

II. Proposed approach

A. Overview

The proposed lossless compression algorithm is based on context-adaptive predictive coding with compressed temporal information. Fig. 1 shows the main building blocks of the proposed encoder. Denoting the index of the current image in the input sequence by n, the previous image (with index $n-1$), also known as the reference frame, is compressed and stored in the on-chip memory. Different algorithms can be selected for compressing the reference frame depending on the application requirements. Compression results using JPEG, JPEG 2000 and JPEG-LS are discussed in Section IV. The decompressed image n–1 is used for both temporal prediction and spatial / temporal mode selection (*Control* block in Fig. 1). To exploit both spatial and temporal higher-order redundancies in the image sequence, context modeling is used in both the spatial path (upper part Fig. 1) and the temporal path (lower part Fig. 1). The *Control* block adaptively selects the spatial or temporal mode based on causal information in the current image and the decompressed previous image. Golomb-Rice code is used to encode the prediction error. In both the spatial and the temporal mode, coding parameter for the Golomb coder is estimated based on statistical information of prediction errors conditioned on the coding context. The spatial path in Fig. 1 resembles the standard JPEG-LS algorithm [10]. The rest of this section focuses on the temporal path, in particular the coding mode selection, the temporal prediction and the temporal coding parameter calculation.

B. Coding mode selection

The coding path in Fig. 1 is adaptively selected by comparing the spatial and temporal variations in the respective spatial and temporal causal neighborhoods of the current pixel. The spatial-temporal variation based adaptive selection has been first proposed in [12]. We use a slightly different approach described as follows. As shown in Fig. 2, the spatial causal neighborhood of the current pixel x contains pixels a, b, c and d, while the temporal causal neighborhood of x contains x', a', b', c' and d'. The spatial variation V_s and temporal variation V_t are calculated as:

$$V_s = |d - b| + |b - c| + |c - a| \tag{1}$$

$$V_t = |a - a'| + |b - b'| + (|c - c'| + |d - d'|)/2 \tag{2}$$

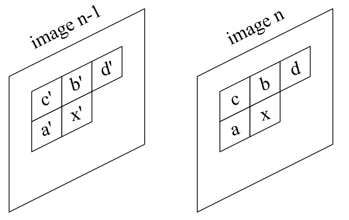

Fig. 2. Pixels in image $n-1$ and n for coding mode selection.

V_s is the sum of absolute values of three causal spatial gradients around x, while V_t is a weighted sum of four causal temporal differences around x. The division by 2 in the last term of (2) is to ensure that V_t scales in the same way as V_s. This hardware-efficient treatment is reasonable in the sense that pixels c and d are geometrically farther away from x than pixels a and b. Spatial path resembling the original JPEG-LS is used if $V_s \leq V_t$, otherwise the temporal path is used.

C. Temporal prediction

Unlike the spatial mode, where the median adaptive predictor [10], [13] calculates the prediction from three fixed predictors as $\hat{x} = \mathit{median}\,(a,\ b,\ a+b-c)$, the temporal mode has only one fixed predictor x'. This is because that if motion information is not available due to complexity- / power-constraints in embedded systems, then the causal neighbor x' from the reference frame is a predictor that should provide the best average results in the temporal mode, especially when the input image sequence has a static background and has few motions in the foreground objects which are small compared with the background (a common case when the capturing device is fixed at a certain distance from the object). The final temporal prediction is refined by temporal context modeling,

$$\hat{x} = x' + C_t(Q_t) \tag{3}$$

where Q_t is the context number determined by quantizing the temporal variation vector $g_t = \{a-a', b-b', c-c', d-d'\}$. A 9-step quantizer $(-\infty, -T_3]$, $[-T_3+1, -T_2]$, $[-T_2+1, -T_1]$, $[-T_1+1, -1]$, $[0]$, $[1, T_1-1]$, $[T_1, T_2-1]$, $[T_2, T_3-1]$, $[T_3, \infty)$ is used to quantize components $a-a'$ and $b-b'$ of g_t, where $T_1 = 3$, $T_2 = 7$, $T_3 = 21$ can be used as default values for 8-bit images, while a 3-step quantizer $(-\infty, -1]$, $[0]$, $[1, \infty)$ is used to quantize components $c-c'$ and $d-d'$. After merging contexts that are symmetric about vector $g_{t_0} = \{0, 0, 0, 0\}$ in a similar way as in JPEG-LS [10], a total of 365 contexts is obtained for the temporal prediction. Procedure for calculating the refinement value $C_t(Q_t)$ is the same as in [10].

D. Temporal coding parameter calculation

Coding contexts used for the Golomb parameter calculation are different from the prediction contexts used in the temporal prediction. Coding contexts are obtained by quantizing V_t by an ad-hoc 5-step quantizer $[0, 1]$, $[2, 6]$, $[7, 20]$, $[21, 49]$, $[50, \infty]$. Using a separate set of contexts for coding may seem contradictory to the goal of a low memory encoder. However it can be readily verified that using such 5 contexts saves 40% context memory (in terms of required bits) compared to using the same context set as used for the temporal prediction, since the most memory-expensive context modeling variable, the sum of magnitudes of prediction errors (register A in [10]), is required by only 5 different contexts compared to 365 in the temporal prediction. Denoting the sum of magnitudes of temporal prediction errors by A_t and the number of context occurrence by N_t, the Golomb parameter k_t for the temporal coding mode is calculated as:

$$k_t = \left\lceil \log_2 \frac{A_t}{N_t} \right\rceil \qquad N_t > 0 \tag{4}$$

The effectiveness of this simple coding context model is verified by our tests using several standard test sequences, where the Golomb parameters based on the 5-context model provide marginally better average coding efficiency than those based on the 365-context model used in the temporal prediction.

III. Memory requirement analysis of different algorithms for reference frame compression

Reference frame compression is a practical option for memory-constrained systems. In principle, the compression algorithm for the reference frame can be either lossy or lossless depending on the application, with lossless compression giving the best temporal prediction. However lossy image compression algorithms can be of higher practical value because of their well known compression efficiency. In this section we select 3 popular lossy image compression algorithms, namely JPEG, JPEG 2000 and near-lossless JPEG-LS, and compare their memory requirements when used in the proposed lossless encoder for image sequences. Experimental results on lossless compression gains versus lossy reference frame compression ratios for these algorithms are reported in Section IV.

Since the encoder scans through the current and the reference frame in raster scan order, and the coding mode selection and prediction discussed in Section II require causal neighbors located in one line above the current pixel, at least one complete line of pixels (not necessarily starting from the border) in the compressed reference frame must be decoded and buffered. It is clear that for transform coding algorithms like JPEG and JPEG 2000, pixels in all blocks (or codeblocks for JPEG 2000) along a row must be decoded and buffered. As can be seen from Table I, although JPEG and JPEG 2000 have better lossy compression performance [4] than JPEG-LS, both of them are more complex [4] than JPEG-LS and need considerably more memory. Specifically, JPEG-LS (as a codec for the reference frame) requires one row of the compressed reference frame

TABLE I
Comparison of JPEG-LS (near-lossless mode), JPEG and JPEG 2000 for reference frame compression

	JPEG-LS	JPEG	JPEG 2000
Required buffer for partially-decoded reference frame	+	++	+++
Lossy compression performance	+	++	+++
Complexity	+	++	+++

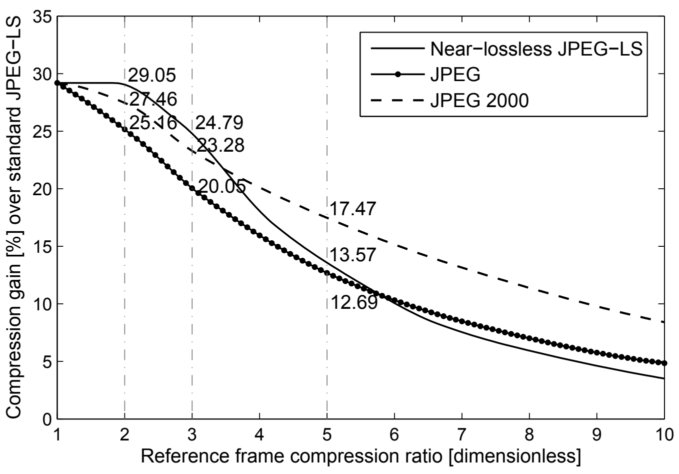

Fig. 3. Compression gain of the proposed encoder over JPEG-LS at different ref frame compression ratios for near-lossless JPEG-LS, JPEG and JPEG 2000

to be decoded and buffered, while JPEG requires 8 times more rows due to the 8×8 DCT, and JPEG 2000 requires e.g. 64×5 rows with a five-level wavelet tree and 64×64 codeblocks. JPEG-LS has both low memory requirement and low computational complexity but worse lossy compression efficiency especially at high compression ratios.

IV. Experimental results

The proposed encoder has been implemented in C++ and its performance has been tested using 5 standard YUV sequences (container$_{352\times288}$, mother-and-daughter$_{352\times288}$, news$_{176\times144}$, salesman$_{360\times288}$, tennis$_{352\times240}$). Only luminance components from the sequences are used. Fig. 3 shows the bicubic-interpolated lossless compression gains over the standard JPEG-LS for different lossy algorithms at different compression ratios of the reference frame for the *container* sequence. It can be seen that for all algorithms, the overall compression gain becomes smaller as stronger compression is applied to the reference frame. Observe that near-lossless JPEG-LS provides better compression gains than JPEG and JPEG 2000 at lower compression ratios e.g 2 and 3. At compression ratios greater than 3.5, JPEG 2000 provides the best compression gain.

Table II shows the compressed file sizes in bits-per-pixel (bpp) and the compression gains over JPEG-LS for all the test sequences with a fixed reference frame compression ratio of 5 (80% reduction of the reference frame buffer size). Specifically, using the computationally efficient near-lossless JPEG-LS leads to 9.6% average gain in lossless compression performance over the standard JPEG-LS.

V. Conclusion

A memory efficient and low complexity lossless encoder for image sequences has been proposed. By compressing the reference frame before storing it, the encoder's memory efficiency has been greatly increased. JPEG-LS with temporal prediction and context modeling is used for inter-frame coding to exploit temporal redundancies in the image sequence. For the reference frame, the performance of different lossy image compression algorithms has been studied using standard test sequences. Experimental results show that using the low complexity near-lossless JPEG-LS for reference frame compression, up to 29% compression gain over the standard JPEG-LS can be achieved by the proposed encoder while reducing half of the memory size compared to storing the uncompressed reference frame. Moreover, using only 20% of memory for the uncompressed reference frame, the proposed encoder is still able to achieve 9.6% average compression gain over JPEG-LS.

TABLE II
Compression gains of the proposed lossless encoder using different algorithms for lossy reference frame compression

Sequence	JPEG-LS (intra-only) [bpp]	Proposed encoder with lossy reference frames					
		JPEG-LS		JPEG		JPEG 2000	
		bpp	gain [%]	bpp	gain [%]	bpp	gain [%]
container	4.35	3.76	13.6	3.80	12.7	3.59	17.5
mother-and-daughter	3.34	3.14	6.0	3.11	6.8	3.07	8.0
news	3.99	3.31	17.1	3.57	10.5	3.28	17.7
salesman	4.38	4.14	5.5	4.02	8.2	4.02	8.2
tennis	5.16	4.85	6.0	4.87	5.6	4.70	8.9
Average	4.24	3.84	9.6	3.87	8.8	3.73	12.1

References

[1] ISO/IEC 15444-1 | ITU-T T.800, *Information technology – JPEG 2000 image coding system – Part 1: Core coding system*, Dec. 2000.

[2] ISO/IEC 15444-3 | ITU-T T.802, *Information technology – JPEG 2000 image coding system – Part 3: Motion JPEG 2000*, Nov. 2001.

[3] ISO/IEC 14495-1 | ITU-T T.87, *Information technology – Lossless and near-lossless compression of continuous-tone still images: Baseline*, Dec. 1999.

[4] D. Santa-Cruz and T. Ebrahimi, "An analytical study of JPEG 2000 functionalities," in *Proc. 2000 IEEE International Conference on Image Processing (ICIP 2000)*, vol. 2, Sep. 2000, pp. 49–52.

[5] G. Schaefer, R. Nowosielski, and R. Starosolski, "Evaluation of lossless image compression algorithms for CFA data," in *ELMAR, 2008. 50th International Symposium*, Sep. 2008, pp. 57–60.

[6] M. Budagavi and M. Zhou, "Video coding using compressed reference frames," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2008)*, Apr. 2008, pp. 1165–1168.

[7] A. Gupte, B. Amrutur, M. Mehendale, A. Rao, and M. Budagavi, "Memory bandwidth and power reduction using lossy reference frame compression in video encoding," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 21, no. 2, pp. 225–230, Feb. 2011.

[8] Y.-Y. Lee, Y.-H. Lee, and T.-H. Tsai, "An efficient lossless embedded compression engine using compacted-FELICS algorithm," in *Proc. 2008 International SOC Conference (SOCC 2008)*, Sep. 2008, pp. 233–236.

[9] P. G. Howard and J. S. Vitter, "Fast and efficient lossless image compression," in *Proc. Data Compression Conference, 1993 (DCC '93)*, Snowbird, Utah, Mar. 1993, pp. 351–360.

[10] M. J. Weinberger, G. Seroussi, and G. Sapiro, "The LOCO-I lossless image compression algorithm: principles and standardization into JPEG-LS," *Image Processing, IEEE Transactions on*, vol. 9, no. 8, pp. 1309–1324, Aug. 2000.

[11] D. Brunello, G. Calvagno, G. Mian, and R. Rinaldo, "Lossless compression of video using temporal information," *Image Processing, IEEE Transactions on*, vol. 12, no. 2, pp. 132–139, Feb. 2003.

[12] K. Yang and A. Faryar, "A contex-based predictive coder for lossless and near-lossless compression of video," in *Proc. 2000 IEEE International Conference on Image Processing (ICIP 2000)*, vol. 1, 2000, pp. 144–147.

[13] S. Martucci, "Reversible compression of HDTV images using median adaptive prediction and arithmetic coding," in *Proc. 1990 IEEE International Symposium on Circuits and Systems (ISCAS '90)*, vol. 2, May 1990, pp. 1310–1313.