Reducing Temporal Redundancy in MJPEG using Zipfian Estimation Techniques

Ngoc-Sinh Nguyen, Duy-Hieu Bui, Xuan-Tu Tran
SIS Laboratory, VNU University of Engineering and Technology (VNU-UET),
144 Xuan Thuy road, Cau Giay, Hanoi, Vietnam. Email: {sinhnn_55,hieubd,tutx}@vnu.edu.vn

***Abstract*—Motion JPEG simply compresses each frame in a video sequence using JPEG still image compression standard. This video codec satisfies power consumption and real-time properties of embedded systems while having very low complexity. However, the downside is its inefficiency in reducing the bit-rate because there are still redundancies between encoded frames. In this paper, we focus on reducing temporal redundancies in MJPEG using a motion detection algorithm, called Zipfian estimation. The Zipfian estimation helps MJPEG extract separately the moving blocks and the stationary blocks then removes temporal redundancies by minimizing encoded bit-stream of stationary blocks and encoding only the residuals of the moving blocks between adjacent frames. Experimental results show that the proposed method can provide twice compression ratio as much as the conventional MJPEG, and an approximate quality and bit-rate as the H.264/AVC (Intra-mode). Even the proposed method uses the Zipfian estimation, it still has smaller number of operations than the conventional MJPEG if the percentage of the static scene is equal or greater than 60%. Compared with the H.264/AVC Intra mode when turning off the rate-distortion optimization, the running time of the proposed method is still only a half.**

I. Introduction

The size of digital videos is the most concerned issue when putting them into limited storage systems and transmitting them through limited bandwidth communication channels. Video compression reduces the size of video without degrading the quality of video into an unacceptable level. This allows more videos to be stored in a given amount of storage systems and requires smaller bandwidth in transmission. The most popular video standard today is the H.264/AVC which achieves 39% bit-rate reduction than the previous standard, the MPEG-4 [1]. Last year, the newest video coding standards, the High Efficiency Video Coding (HEVC) [2] has been released and can achieve even 50% bit-rate reduction than the H.264/AVC. However, these standards are too complex and therefore, they are not suitable for embedded systems with limited resources.

MJPEG is a purely spatial video compression using the JPEG standard. It compresses independently each frame of video into JPG format. Compared to the H.264/AVC, the MJPEG requires lower CPU performance and has higher processing speed that satisfies power consumption and real-time properties of embedded systems. However, the size of encoded video using MJPEG is much larger than using the H.264/AVC, which raises the requirements of the storage space and network bandwidth.

JPEG is the most popular image compression standard [3] thanks to its simplicity and efficiency. JPEG removes spatial redundancies that are low effect on human's eyes system. Figure 1 shows the main processes in JPEG encoder lossy form. It includes forward Discrete Cosine Transform (FDCT), Quantization, Zigzag Ordering, Entropy Encoding. The FDCT converts pixel values into frequency domain. The transformed coefficients are quantized using quantization table to remove spatial redundancies, and read in a zigzag scan. Finally, the zigzag sequence is encoded by Entropy Encoding.

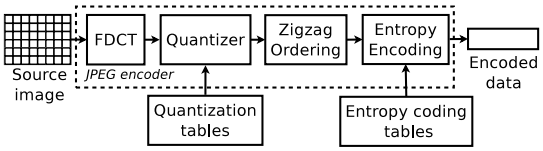

Fig. 1. JPEG encoder.

In this paper, to improve the compression ratio of the conventional MJPEG while keeping its simplicity, we applied a low complexity but robust motion detection algorithm, called Zipfian estimation [4], [5], to the MJPEG. We focus on reducing temporal redundancy in MJPEG using Zipfian estimation techniques. As the conventional MJPEG, the proposed codec encodes frames into JPG bit-stream but instead of encoding original frames, the codec encodes only the residual of motions and minimizes encoded bit-stream of stationary blocks.

The rests of the paper are organized as follows. Section II introduces Zipfian estimation. Section III describes the proposed codec architecture. Section IV presents the experimental results. Finally, conclusions and remarks will be provided in Section V.

II. Zipfian Estimation

Zipfian estimation is the one of the fastest motion detection algorithms [4], [5], see Algorithm 1. It is improved from basic $\Sigma - \Delta$ background subtraction algorithm [6]. The algorithm builds a model of static scene, called background, then analyzes the difference between current frame and the background to get the moving pixels, called foreground. Firstly, the algorithm computes a threshold according to the frame index t: p is the value of the index module 2^m (m is the number of bits representing a pixel). π is the value of the greatest power of 2 that divides p. Then, the threshold is equal 2^m divided by π. The background M_t is updated whenever variance V_t is greater than σ. Next, O_t is the absolute difference between I_t and M_t. To avoid auto-reference, the variance V_t is updated using

a constant period Tv (usually the power of 2 and in the range from 1 to 64). N is an amplification factor of the variance V_t (usually from 1 to 4). Finally, moving or stationary of a pixel is decided by comparing the absolute difference to the variance. As shown in Algorithm 1, the Zipfian estimation works on any size fixed-point arithmetic using only comparison, increment and absolute difference. The work in [7] indicates that Zipfian is much faster than the others.

Algorithm 1: Zipfian estimation [6]

find *the greatest* 2^p *that divides (t mod* 2^m*)*
set $\sigma = 2^m/2^p$
foreach *pixel* x **do**
  if $V_{t-1}(x) > \sigma$ **then**
    if $M_{t-1}(x) < I_t(x)$ **then** $M_t(x) \leftarrow M_{t-1} + 1$
    if $M_{t-1}(x) > I_t(x)$ **then** $M_t(x) \leftarrow M_{t-1} - 1$
foreach *pixel* x **do**
  $O_t(x) = |M_t(x) - I_t(x)|$
foreach *pixel* x **do**
  if t *mod* $T_V = 0$ **then**
    if $V_t(x) < N \times O_t(x)$ **then** $V_t(x) \leftarrow V_{t-1}(x) + 1$
    if $V_t(x) > N \times O_t(x)$ **then** $V_t(x) \leftarrow V_{t-1}(x) - 1$
foreach *pixel* x **do**
  if $O_t(x) < V_t(x)$ **then** $E_t(x) \leftarrow 0$

III. The Proposed Video Codec Architecture

As mentioned above, in this work, we improve the compression ratio of the conventional MJPEG by applying the Zipfian motion estimation algorithm. Therefore, the Zipfian motion estimation algorithm has been combined with the JPEG standard to reduce not only spatial redundancy but also temporal redundancy in video coding. The proposed encoding architecture is shown in Figure 2. The proposed encoder has three main tasks: detecting 8×8 motion blocks, JPEG encoding the moving blocks and the stationary blocks, and reconstructing the residuals for the next frame.

The first task consists of the following processes: *Zigzag block ordering*, *Motion detection*, and *Extracting moving block*. The *Zigzag block ordering* process rearranges the block processing order of the luma component as in Figure 3, while the *Motion detection* process detects the moving pixels and computes the sum of the moving pixels in each block by Zipfian estimation of the luma component. If the sum of the moving pixels in a block is equal to or greater than a threshold β which is the minimum number of the moving pixels per block (usually from 8 to 24), this block will be assigned to 1 (temporary moving block), otherwise 0 (temporary stationary block). Next, the *Extracting moving block* process works as a filter. It discards the single marked 1 blocks and creates a border around the moving zones by considering the relationship between blocks. Algorithm 2 shows the *Extracting moving block* process in the gray channel. The types of blocks in the color channels depend on the corresponding blocks in the gray channel of the MCU.

Algorithm 2: Extracting moving blocks

Set *a constant to define the moving or stationary block*
$const = 2$
Find *the number of the temporary moving neighbors of the current blocks* x
$sum = 0$
foreach *neighbor of block x* **do**
  if *neighbor = "temporary moving"* **then**
    $sum+ = 1$
if $sum > const \mid \{x = 1 \;\&\&\; sum > 0\}$ **then**
  x=moving
else
  x=stationary

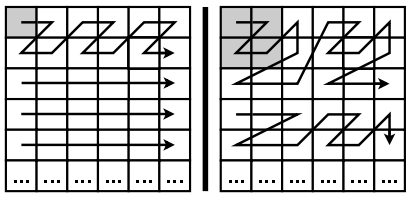

Fig. 3. Zigzag block order at downsampling: Y only (left), YUV:420 (right).

JPEG compression is the main algorithm in the second task. For the still blocks, all the pixel values are assigned to zero in frequency domain. Hence, those blocks are directly encoded by *Entropy encoding*. The encoded bit-stream of the still blocks is only the encoded DC values and the End-Of-Block (EOB) symbols. For the moving blocks, instead of encoding them directly, we encode the residuals of the moving blocks. The residuals D_t are the difference between the current frame I_t and the previous decoded frame I'_{t-1} (Equation 1). This helps to reduce more bit-rate as in the H.264/AVC.

$$D_t(x) = \frac{I_t(x) - I'_{t-1}(x)}{2} \tag{1}$$

Finally, in the third task, the transformed residuals are reconstructed to create the reconstructed frame I'_{t-1}. The reconstructed frame will be used as a reference frame for the jpeg encoding process of the next frame.

Our proposed encoder has two more tasks in addition to the original MJPEG, however, a large number of operations are saved in forward DCT and quantization processes for the still blocks. Encoding the residuals of the moving blocks reduces even more bit-rate than encoding them directly. The encoded bit-stream of still blocks is minimized, it only contains the encoded DC values and the EOB symbols.

$$I'_t(x) = 2 \times D_t(x) + I'_{t-1}(x) \tag{2}$$

The architecture of the decoder is much simpler than the encoder, as depicted in Figure 4. Based on the information from *Inverse entropy encoding*, the still blocks are detected by the zero DC values and the EOB symbols. For each still block, the pixel values are directly copied from the

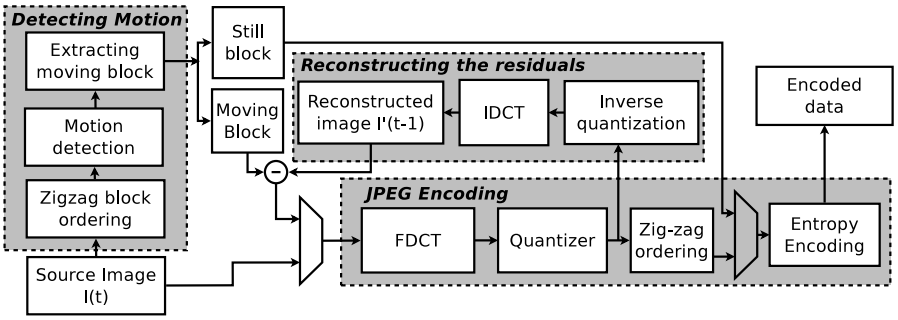

Fig. 2. The proposed encoding architecture.

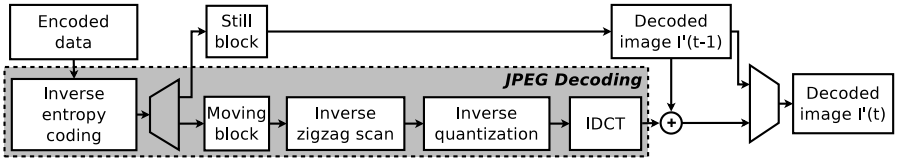

Fig. 4. The proposed decoding architecture.

reconstructed frame I'_{t-1}. For the moving blocks, after being decoded using JPEG algorithm, the pixel values will be rebuilt using Equation 2. The proposed decoding architecture is much faster than the default JPEG decoder because we save many operations in *Inverse quantization* and *Inverse DCT* processes for the still blocks.

IV. Experimental Results

The proposed codec has been implemented using SystemC/C++ programming language – a system-level modeling language. The testbench reads raw videos and encodes frames into JPG files. In our project, the bit-rate, the quality and the complexity of the proposed codec have been compared to the original MJPEG and H.264/AVC. The test sequences include Container, Hall, Foreman, and News [9] in CIF resolution. Figure 5 shows the test results with Hall sequence. The first row is the original frames, the second row is the encoded frames, and the last row is the decoded frames at 1^{st}, 75^{th}, 230^{th} frames, respectively.

Fig. 5. Hall sequence results.

The comparison of the compression ratio between the proposed codec and the conventional MJPEG is shown in Figure 6. In this test, we use the JPEG compression model [8] for MJPEG. The quantization parameter of the JPEG compression model is changed from 5 to 100. In our proposed codec, $N = 2$, $Tv = 4$ and $\beta = 16$ are fixed values in *Detecting motion* process. Figure 6 shows that our proposed codec provides better/larger compression ratio range than the conventional MJPEG. The proposed codec can achieves more than 60:1 compression ratio while MJPEG gets less than 40:1 compression ratio. At the same PSNR, the proposed codec has much higher compression ratio than the conventional MJPEG. For example, with PSNR at $30dB$, the compression ratio in the conventional MJPEG is only about 20:1 while the proposed codec is more than 40:1.

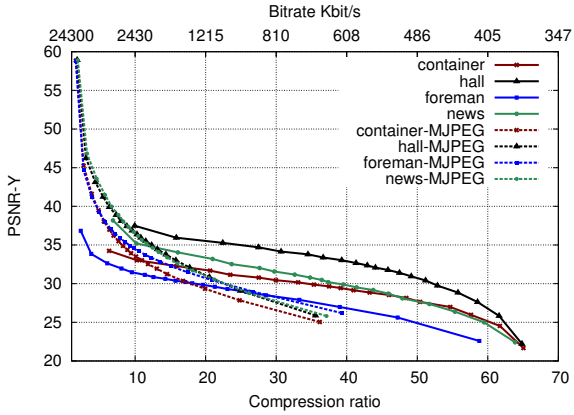

Fig. 6. The comparison between the proposed method and MJPEG.

Figure 7 presents the comparison of the bit-rate and PNSR between our proposed codec and the reference software of H.264/AVC: JM [10] with only Intra prediction mode. The proposed codec gets an approximate PSNR at the same bit-rate as the H.264/AVC Intra mode. With videos from stationary cameras such as News and Hall sequences, our codec has higher PNSR than the H.264/AVC Intra mode.

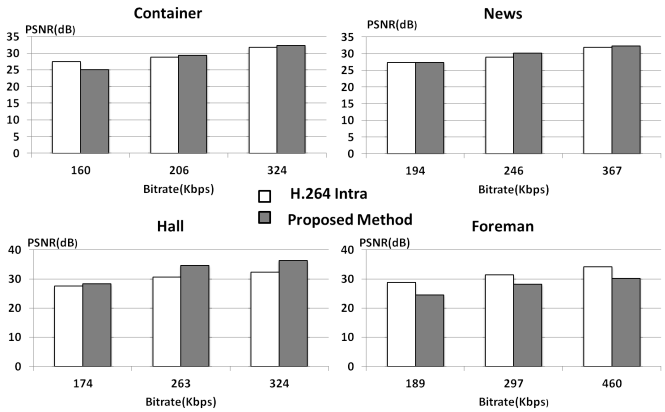

Fig. 7. The comparison between the proposed method and the H.264/AVC Intra mode.

The running time of the Hall sequence using the proposed encoding method and the JM H.264/AVC Intra mode with rate-distortion optimization (RDO) 'on' and 'off' is shown in Figure 8. The test system uses an Intel(R) Core(TM) i3 M 370 CPU at $2.40GHz$. At the same bit-rate, running time of the proposed codec is haft the runing time of the JM H.264/AVC Intra mode when RDO is off. The running time of the H.264/AVC Intra mode, when RDO is on, is 6 times longer than the one of our method at $1500kbps$. The proposed codec has smaller bit-rate range than the H.264/AVC Intra mode because of the limitation of the MJPEG algorithm.

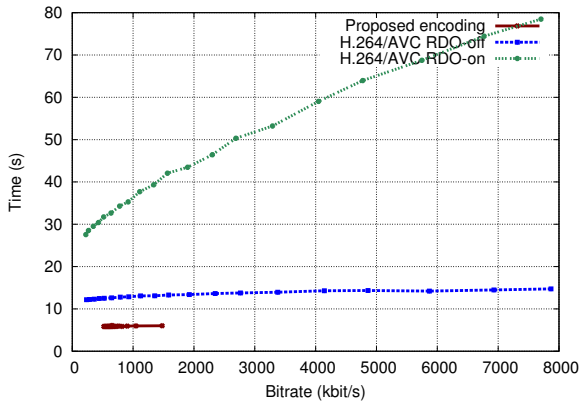

Fig. 8. Comparison bit-rate and running time with JM H.264/AVC.

Table I shows the number of operations used in Zipfian estimation at $Tv = 4$, forward DCT and quantization. The number of operations in Zipfians is much smaller than the one in FDCT and quantization. It is about 1/5 as small as the number of operations in FDCT and Quantization. This means that detecting motion of 5 blocks using Zipfian estimation is still simpler than doing FDCT and quantization of one block in MJPEG. Obviously, the proposed codec will has the smaller number of operations than MJPEG if the percentage of the still blocks is greater than 60%. Figure 9 shows the average percentage of the still blocks in some sequence with $N = 2$ and $N = 3$. In the tested sequences, the percentage of still blocks is greater than 60% except for the Foreman sequence.

TABLE I
THE NUMBER OF COMPUTATIONS PER BLOCK FOR ZIPFIAN ESTIMATION, FDCT AND QUANTIZATION IN OUR CURRENT IMPLEMENTATION

Operation	Zipfian	FDCT	quantization
Addition	200	928	64
Multiplication	0	192	64
Shift	20	128	0

This is because the Foreman sequence is recorded from a moving camera.

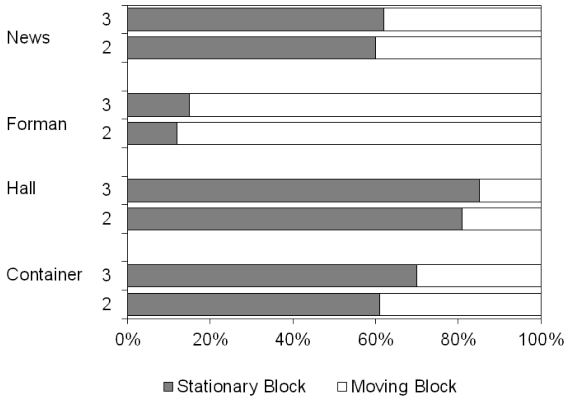

Fig. 9. Average percentage of still blocks in some sequences.

V. CONCLUSIONS

We presented the proposed encoding/decoding architecture to improve the quality and bit-rate of the conventional MJPEG by combining JPEG image compression and Zipfian estimation. The proposed codec helps the MJPEG remove temporal redundancies by the minimization of bit-stream for stationary blocks and encoding only the residuals of the moving blocks. The experimental results show that our proposed codec gets much higher compression ratio than the conventional MJPEG and approximately equal to the JM H.264/AVC Intra Mode at the same PSNR but with lower complexity.

ACKNOWLEDGMENT

The authors would like to thank Nafosted for travel grant.

REFERENCES

[1] A. Joch and et al., "Performance comparison of video coding standards using Lagragian coder control," in *IEEE ICIP*, 2002, pp. 501–504.
[2] *Recommendation ITU-T H.265/HEVC*, ITU-T Std., Rev. 1.0, April 2013.
[3] *ITU-T Recommendation T.81*, The International Telegraph and Telephone Consultative Commitee, September 1992.
[4] A. Manzanera, "Sigma-delta background subtraction and the zipf law," *CIARP.LNCS*, vol. 28-2, pp. 42–51, 2007.
[5] A. Manzanera and L. Lacassagne, "Motion detection: fast and robust algorithms for embedded system," *16th IEEE Conf. on Image Processing (ICIP)*, pp. 3265–3268, 2009.
[6] A. Manzanera and J. C. Richefue, "A robust and computationally efficient motion detection algorithm based on sigma-delta background estimation," *ICVGIP. IEEE*, pp. 46–51, 2004.
[7] O. Barnich and M. Droogenbroeck, "ViBe: A universal background subtraction algorithm for video sequences," *Image processing, IEEE Transactions*, vol. 20, pp. 1709–1724, 2011.
[8] "JPEG compressor open source code," http://code.google.com/p/jpeg-compressor/.
[9] "Video sequences," http://media.xiph.org/video/derf/.
[10] "JM H.264/AVC," http://iphome.hhi.de/suehring/tml/.