THE FAST CLOSE-LOOP VIDEO TRANSCODER WITH LIMITED DRIFTING ERROR

Lujun Yuan[1], Feng Wu[2], Qi Chen[2], Shipeng Li[2], Wen Gao[1]

[1]Institute of Computing Technology, Chinese Academy of Science, Beijing
[2]Microsoft Research Asia, Beijing
{ljyuan, wgao}@jdl.ac.cn, {fengwu, i-qichen, spli}@microsoft.com

Abstract

This paper proposes a fast close-loop video transcoder with limited drifting error. Firstly, similar to the traditional close-loop transcoder, the proposed transcoder still accumulates the re-quantization errors of each I or P picture. However, the accumulated errors are not always introduced into the transcoding loop in every block. A triple-threshold algorithm is proposed to adaptively utilize the accumulated errors at block level so as to control drifting error under an acceptable level. Secondly, when the accumulated errors are not introduced into the transcoding loop, additional DCT transform in the proposed transcoder can be removed. Furthermore, the re-quantization process can be simply implemented by looking up table, which significantly reduces the complexity of the re-quantization process. The experimental results show that picture quality of the proposed transcoder is closed to that of the traditional closed-loop one while about 40% blocks are not updated with the accumulated errors. In this case, the transcoding process can be speeded up to 30%.

1. Introduction

With the maturity and extensive deployment of video coding technologies, most of digital video contents exist in the coded form (e.g. MPEG-2 or other) for saving storage. Transcoding the coded video contents from one format to another, from one bit rate to another and from one solution to another is indispensable in many multimedia applications, such as digital video broadcasting, video on demand (VOD), teleconferencing and long distance learning, and so on [1]~[8]. This paper mainly investigates how to fast transcode MPEG-2 streams from a high bit rate to a desired low bit rate.

In the literature, the bit-rate reduction transcoding problem has been addressed in [1]~[8]. The straightforward way to do video transcoding is to cascade a decoder and an encoder together. It first performs a full decoding of the input stream and then performs a full encoding of the decoded video sequence. Since the encoder can optimize video quality at the target bit rate by estimating another set of motion vectors and re-allocating bits of each picture, the cascaded transcoder typically achieves better video quality. However, the complexity is very high in this case. As a matter of fact, information in the original stream, such as motion vectors, macroblock modes, rate control information can be borrowed to significantly accelerate the video encoding. As a result, the open-loop and close-loop architectures are proposed to implement video transcoding.

The open-loop video transcoder just re-quantizes DCT coefficients of the original bitstream to achieve the desired lower bit rate without taking modified references into account [1]. However, the reconstructed references in the transcoded stream are different from that used in the original stream. Such mismatches would bring so-called drifting error. It not only affects the quality of the current picture but also is propagated to subsequent pictures until the next I picture. In order to reduce drifting error in video transcoding, the close-loop video transcoder was proposed [3][7]. By using motion vectors in the original stream, it can merge the decoding and encoding loops in the cascaded transcoder together. So the drifting errors can be accumulated in the prediction loop and used to compensate the re-quantization errors during the transcoding process.

This paper proposes a fast MPEG-2 video transcoder from the close-loop one by allowing limited drifting error. The proposed transcoder still accumulates the re-quantization errors of each I or P picture. However, unlike the traditional close-loop one, the accumulated errors are not always introduced into the transcoding loop. A triple-threshold algorithm is proposed to adaptively utilize the accumulated errors at block level so as to control drifting error under an acceptable level. Secondly, when the accumulated errors are not introduced into the transcoding loop, additional DCT transform in the proposed transcoder can be removed. Furthermore, this paper also proposes a novel implementation of the re-quantization process by looking up table, which significantly simplifies the quantization and de-quantization operations.

The rest of paper is organized as follows. Section 2 first analyzes the quantization process in the traditional close-loop transcoder and then proposes a fast transcoder with limited drifting error. In this section, the triple-threshold algorithm and the implementation of re-quantization are also discussed in detail. Experimental results are given in Section 3. Finally, Section 4 concludes this paper.

2. Proposed fast MPEG-2 transcoder

Although the close-loop transcoder significantly reduces the complexity of the transcoding process, it is always trying to compensate drifting error by introducing the accumulated errors into the transcoding loop. Obviously, this would increase the complexity of the re-quantization process. So a question is arising here whether we can allow limited drifting error to further reduce the complexity of the close-loop transcoder.

The quantization process in the close-loop transcoder is sketchily represented by using an integer division.

$$Y = (X + E) / q \tag{1}$$

X is the de-quantized DCT coefficient from the original stream. E is the accumulated error after DCT transform. q is the step size of this quantization. Y is the re-quantized DCT coefficient. If the accumulated error E is subject to

$$E < q - X\%q \tag{2}$$

the non-zero E does not affect the value of Y. In other words, although the accumulated error is always introduced into the transcoding loop in the traditional close-loop transcoder, it only takes effect when E is more than a certain threshold.

Therefore, we proposed a new close-loop transcoder as shown in Figure 1, where one switch is proposed to intentionally control whether or not to update an 8x8 block with the accumulated errors. Here, TH is a threshold. If the accumulated error in a block is more than the threshold, it is introduced into the transcoding loop as in the traditional close-loop transcoder; otherwise input non-zero levels will simply go through the re-quantization process. In this case, DCT transform to the accumulated error in Figure 1 is not necessary. The larger the threshold, the simpler the transcoding, but it could also bring more loss in picture quality.

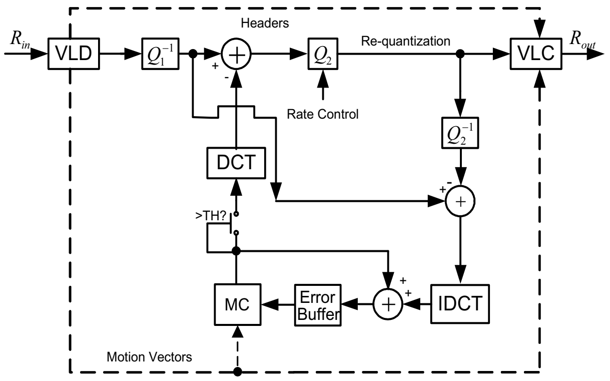

Figure 1: The block diagram of the proposed transcoder.

Generally, the drift-free process in the close-loop transcoder consists of two processes: error accumulation and error compensation. Since B pictures in MPEG-2 are never used as reference, the drift error in B pictures can not be propagated to other pictures. So there is no error accumulation process for B pictures. Furthermore, even if the accumulated errors are not introduced into the transcoding loop in B pictures, it would not result in severe quality lose. Thus in the proposed transcoder blocks in B pictures are not updated with the accumulated errors. In fact, the structure for transcoding B pictures is same as the open-loop one. Since both I pictures and P pictures will be used as reference for other pictures, it is necessary to update the Error Buffer in Figure 1 with the accumulated errors. However, because I pictures and intra blocks in P pictures are coded without any prediction from other pictures, no drift error compensation is need for them. At last, inter blocks in P pictures will be dealt with the following proposed method.

2.1 Proposed triple-threshold algorithm

This sub-section proposes a triple-threshold algorithm to control whether or not to update an 8x8 block with the accumulated errors. The basic idea is that the threshold of each block is dynamically selected along with the counter of a block.

```
switch (block_counter)
{
    case 0:    TH = TH1; break;
    case 1:    TH = TH2; break;
    default:   TH = TH3; break;
}

$s = \sum_{i=0}^{7}\sum_{j=0}^{7}\left|e(i,j)\right|$;

if (Intra block)
    block_counte=0;
else if (Inter block in P picture)
{
    if ($s > TH$)
    {
        drifting error compensation;
        if (block_counter > 0)
            block_counter = block_counter – 1;
    }
    else
    {
        no drifting error compensation;
        block_counter = block_counter + 1;
    }
}
```

Figure 2: The pseudo-code of the proposed algorithm.

Firstly, the accumulated error of each block is measured with the sum of absolute error s, i.e,

$$s = \sum_{i=0}^{7}\sum_{j=0}^{7}\left|e(i,j)\right| \tag{3}$$

e(i, j) is the accumulated error of each pixel in a block. Here are three thresholds *TH1*, *TH2*, and *TH3*, also, *TH1* > *TH2* > *TH3*. The larger the threshold, the less likelihood blocks are compensated with the accumulated error, then the simpler the transcoding process, but it could also bring more loss in picture quality.

In order to achieve a better trade-off between transcoding complexity and picture quality, the selection of the threshold is described with pseudo-code as in Figure 2, where every block has a unique counter variable *block_counter*. The larger *block_counter*, the less the selected threshold, then the more likelihood blocks are compensated with the accumulated error. *block_counter* is set as zero when it is an intra block. When *block_counter* is 0, i.e., drifting error propagated from previous pictures is zero or very small, the large threshold *TH1* is selected. When *block_counter* is 1, the moderate threshold *TH2* is selected; otherwise the small threshold *TH3* is selected. If an inter block in P picture is compensated with the accumulated errors and its *block_counter* is more than 0, *block_counter* decreases 1. If an inter block in P picture is not compensated with the accumulated errors, *block_counter* increases 1.

2.2 Re-quantization with looking up table

When the accumulated errors are not introduced into the transcoding loop or block is an intra one, decoded non-zero levels will simply go through the re-quantization process, which is separately depictured in Figure 3. These de-quantization and quantization modules that are normally required by each transcoding process cost a great deal of CPU cycles. In this sub-section, we propose a simple implementation of re-quantization by looking up table technique.

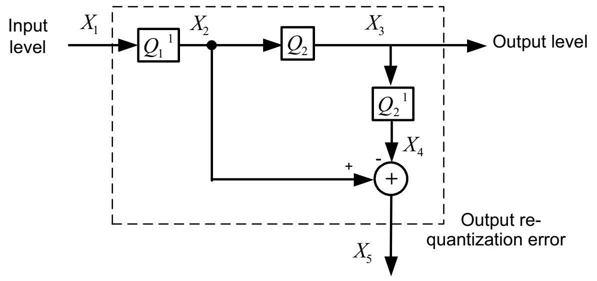

Figure 3: The traditional re-quantization process.

In Figure 3, X_1 is the decoded level which is extracted from the original video bitstream by variable length decoding (VLD); X_2 is the DCT coefficient which is obtained by the de-quantization process with the original quantization parameter q_1; the output level X_3 is calculated by the second quantization process with the new parameter q_2. In addition, for accumulating the re-quantization error, the second de-quantization DCT coefficient is given as X_4; the re-quantization error is given as X_5.

The output level X_3 and the re-quantization error X_5 can be formulated as follows.

$$X_3 = F_1(X_1, q_m, q_1, q_2) \quad (4)$$

$$X_5 = F_2(X_1, q_m, q_1, q_2) \quad (5)$$

Here q_m is the corresponding element in the MPEG-2 quantization matrix; $F_1()$ consists of the de-quantization and quantization process; $F_2()$ consists of the de-quantization, quantization and de-quantization process.

Obviously, $F_1()$ and $F_2()$ can be simply implemented by looking up table with input parameters X_1, q_m, q_1, q_2 and output results X_3 and X_5. For the default intra quantization matrix, q_m has 19 possible values. The quantization parameter q_1 or q_2 varies from 1 to 31. However, because of bit-rate reduction transcoding, it is reasonable to assume $q_2 \geq q_1$, so there are 31×16 possible combinations of q_1 and q_2. The input decoded level X_1 is in the range [1, 2047]. So the intra table consists of 31×16×19×2047 elements. Each element has 4 bytes with 2 byte for X_3 and 2 bytes for X_5. It would cost about 77M bytes. For the default inter quantization matrix. q_m is always equal to 16. So the inter table costs about 4M bytes. In MPEG-2, the quantization matrix can be defined by user instead of the default ones. In the worst case, q_m may have 64 different values. The table size will be far more than the previous estimation. Either software or hardware way is not easy to implement such large tables. Furthermore, the large size table also affects the speed of memory access because CPU usually has a small cache. Therefore we should shrink the table sizes.

Firstly, more than 90% X_1 locates at [1, 40]. When X_1 is more than 40, the re-quantization process is not implemented by looking up table so as to reduce the table sizes. Secondly, we can approximately calculate X_3 and X_5 as follows,

$$X_3 = X_3' = F_1(X_1, 16, q_1, q_2) \quad (6)$$

$$X_5 = \frac{X'_5 \times q_m}{16} = \frac{F_2(X_1, 16, q_1, q_2) \times q_m}{16} \quad (7)$$

X'_3 and X'_5 are obtained by looking up table. Although equations (6) and (7) introduce errors, we can not observe its effects in term of PSNR value and picture quality. By the above two improvements, the size of intra and inter table is about 79K bytes. Furthermore, it dose not need to re-calculate the tables when quantization matrixes are change. It is very important especially in hardware implementation.

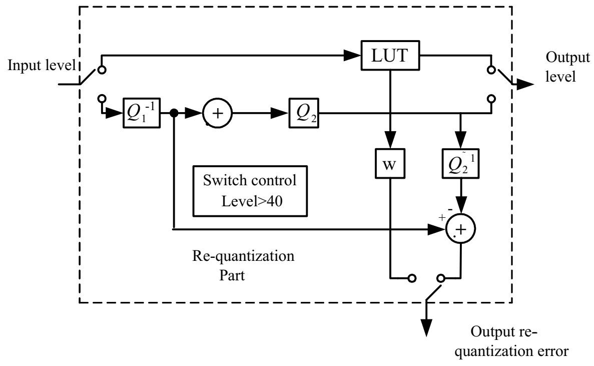

Figure 4: The Re-quantization by lookup tables.

Figure 4 shows the proposed implementation of the re-quantization process. There is a switch in the block diagram, which is controlled by input level value. When input level is more than 40, the re-quantization is performed as the original way; otherwise it is simply implemented by looking up table. Module **w** indicates the multiply and shift operations.

3. Experimental results

In our experiments, the test video sequence is extracted from one movie trailer. The resolution of the sequence is 720×480, and the frame rate is 23.976. Firstly, this sequence is coded with MPEG2 encoder, its bit rate is set to 10 Mbps. Then this 10Mbps video bitstream is used as the original video bitstream for our transcoding experiments. In order to evaluate our proposed transcoding method, three transcoding methods are implemented: open-loop architecture (OPEN), close-loop architecture (CLOSE) and our proposed architecture (FAST). Notice that in order to compare with our proposed transcoder, no drifting error compensation for B-pictures in the close-loop transcoder.

Table 1: The performance comparison of three transcoders

QP		Bit-rate (Mbps)	PSNR(dB)	Speed (fps)
4	OPEN	6.8424	43.1767	44.4972
	CLOSE	6.9873	43.5035	37.1747
	FAST	6.9406	43.4828	46.0440
8	OPEN	3.1942	40.4151	52.0879
	CLOSE	3.3435	41.1372	41.3793
	FAST	3.3081	40.8459	52.8961
12	OPEN	2.2100	38.8437	54.8296
	CLOSE	2.2768	39.5277	44.0885
	FAST	2.2739	39.4180	56.1430
16	OPEN	1.6288	37.5404	56.6572
	CLOSE	1.6908	38.2615	45.2318
	FAST	1.6742	37.9097	58.5366
20	OPEN	1.3954	36.5598	57.8983
	CLOSE	1.4459	37.2822	46.0440
	FAST	1.4159	37.0797	59.2593

We transcode the original video bitstream with fixed quant parameter value (QP): 4, 8, 12, 16 and 20. The results are listed in table 1. Notice that in order to comparison fairly, we implement three transcoders without any CPU instruction level optimization. And for our propose transcoder, about 40% blocks in P-pictures without drifting error compensation, the PSNR is less about 0.2dB than the close-loop one, and the speed is higher about 30% than the close-loop one. Furthermore, the proposed scheme is faster even than the open-loop one because of looking up table.

4. Conclusions

In this paper, a novel low-complexity transcoding method for bit-rate reduction of MPEG-2 bitstream is proposed. The proposed low complexity transcoding scheme is based on the closed-loop architecture. The update switch is added to control whether to perform the drift error compensation for P-type pictures, and the loop-up table technique is used to speed up de-quantization and re-quantization processing. Because of these two techniques, the transcoding speed of MPEG-2 to MPEG-2 bit-rate reduction video transcoder dramatically increase compared with the closed-loop architecture, and the video quality is closed to the closed-loop architecture, but better than the open-loop architecture. Furthermore, this proposed scheme is very useful in complexity-scalable video transcoding by adjusting the threshold, where the video transcoder should adaptively select appropriate transcoding method according host CPU utilization. Therefore, the proposed low-complexity video transcoder fits in this CPU-awareness application.

References

[1] H. Sun, W. Kwok, and J. Zdepski, “Architectures for MPEG compressed bitstream scaling”, IEEE Trans. Circuits Syst. Video Technol. vol. 6, pp. 191-199, 1996.

[2] J. Youn, M. T. Sun, and C. W. Lin, “Motion vector refinement for high-performance transcoding”, IEEE Trans. Multimedia, vol. 1, pp. 30-40, 1999.

[3] H. J. Stuttgen, “Network evolution and multimedia communication”, IEEE Multimedia, vol. 2, pp. 42-59, 1995.

[4] Q. Eleftheriadis and D. Anastassiou, “Meeting arbitrary QoS constraints using dynamic rate shaping of coded digital video”, 5th Int. Workshop Network and Operating Syst. for Digital Audio and Video, pp. 95-106, 1995.

[5] Y. Nakajima, H. Hori and T. Kanoh, “Rate conversion of MPEG coded video by re-quantization process”, IEEE Int. Conf. Image Processing, vol. 3, pp. 408-411, 1995.

[6] P. Assuncao and M.Ghanbari, “Post-processing of MPEG-2 coded video for transmission at lower bit rates”, IEEE Int. Conf. Acoust., Speech, Signal Processing, vol. 4, pp.1998-2001, 1996.

[7] P. Yin, A. Vetro, B. Liu, and H. Sun, “Drift compensation for reduced spatial resolution transcoding”, IEEE Trans. Circuits Syst. Video Technol., vol. 12, pp. 1009-1020, 2002.

[8] P.Assuncao and M.Ghanbari, “A frequency-domain video transcoder for dynamic bit-rate reduction for MPEG-2 bitstreams”, IEEE Trans. Circuits Syst. Video Technol., vol.8, pp.953-967, Dec.1998.