Efficient Block Size Selection for MPEG-2 to H.264 Transcoding

Gao Chen Institute of Computing Technology, Chinese Academy of Sciences Beijing, 100080, P.R.China 86-10-8261-1846 chengao@ict.ac.cn

Yong-dong Zhang Institute of Computing Technology, Chinese Academy of Sciences Beijing, 100080, P.R.China 86-10-8261-1846 zhyd@ict.ac.cn

Shou-xun Lin Institute of Computing Technology, Chinese Academy of Sciences Beijing, 100080, P.R.China 86-10-8261-1846 sxlin@ict.ac.cn

Feng Dai Institute of Computing Technology, Chinese Academy of Sciences Beijing, 100080, P.R.China 86-10-8261-1846 fdai@ict.ac.c

ABSTRACT

In this paper, an efficient block size mode selection algorithm for the variable-sizes block-matching (VSBM) in the MPEG-2 to H.264 transcoding is presented. Depending on leveraging the available motion information carried by the MPEG-2 bit-streams, the proposed algorithm is used to determine which one of the 16x16, 16x8, 8x16, and 8x8 block size modes should be used for each macroblock (MB). The simulation results show that the performance of the proposed algorithm is close to that of a cascaded pixel-domain transcoder (CPDT) when all the seven block size modes are enabled and the exhaustively full search method is used to determine the best block size modes. The whole transcoding time can be efficiently reduced by 22% on the average while the bit rate is slightly increased (2.9%).

Categories and Subject Descriptors

I.4.2 [Image processing and Computer Vision]: Compression (Coding) -Transcoding, Motion Estimation.

General Terms

Algorithms, Design, Experimentation.

Keywords

Video Transcoding, H.264, MPEG-2, Mode Decision

1. INTRODUCTION

The new generation video standard H.264 [1], which can achieve a major improvement in the rate-distortion efficiency providing typically a factor of two in bit-rate saving comparing to MPEG-2 [2], is expected to replace the use of MPEG-2 in digital video systems. However, considering the fact that MPEG-2 has

MM'04, October 10-16, 2004, New York, New York, USA.

Copyright 2004 ACM 1-58113-893-8/04/0010...\$5.00.

been successfully used in many applications including DVD and digital TV broadcast, the complete migration to H.264 will take several years. MPEG-2 to H.264 transcoder can be used to leverage the compression efficiency offered by H.264 with broadcast quality content produced in MPEG-2 format. Adopting H.264 will require transcoders to convert MPEG-2 to/from this newly emerging standard H.264 as necessary [3].

The JVT reference software [4], adopts full searching process (the examination of all the seven block size modes) for motion estimation (ME) and motion compensation (MC) on a Rate Distortion Optimization (RDO) framework. This full searching process provides the best coding result but the increase in computation is linearly with the number of block size modes used [1]. So, in the MPEG-2 to H.264 transcoding, the H.264 reencoding process is the most critical part in terms of computational complexity. The computational complexity can be reduced by leverage the motion information carried in the MPEG-2 bit-streams. In this paper, we focus on exploiting the MPEG-2 16x16 mode motion estimation results to determine whether it is necessary to further search more block size modes.

The rest of the paper will be organized as follows: section 2 provides some observation and analysis on the statistics of the energy of MPEG-2 residual MB. Section 3 describes the proposed fast multi-block selection algorithm and experimental result is presented in section 4. We close the paper with concluding remarks in section 5.

2. OBSERVATIONS AND ANALYSIS ON THE ENERGY OF MPGE-2 RESIDUAL MB

The simplest way to transcode on video is directly cascading a source video decoder with a destination video encoder, i.e. the CPDT architecture transcoder. Obviously, this direct approach is usually computationally intensive and represents the upper bound on the rate-distortion performance of the transcoded video [3].

In MPEG-2 to H.264 transcoding, many characteristics between MPEG-2 and H.264 are different, such as coding methods, motion mode definitions and picture coding types. Motion information contained in MPEG-2 video cannot be used directly. However, after analyzing and modifying the extracted motion information, transcoders can still explore them to make

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

transcoding process more efficiently. In this paper, we adopt a general architecture for MPEG-2 to H.264 video transocding, as shown in Figure 1, consists of two stages: a MPEG-2 decoding stage that converts the compressed input MPEG-2 bit-stream to an uncompressed frame representation of the input, followed by a H.264 re-encoding stage that encodes the input to H.264 video format bit-streams. Retrieved motion information, gathered during the first stage, is often used as an aid for the second stage.



Fig.1. MPEG-2 to H.264 transcoding architecture

It is well known that the performance of inter mode is measured by the energy of residue. The measure of the energy of residual MB we used is the sum of the absolute value of the dequantized DCT coefficients of the motion compensated prediction MPEG-2 residual MB. Let the variable E be defined as the energy of the MPEG-2 residual MB. The function Prob₁(V₁ $\leq E < V_2$) denotes the MPEG-2 residual MB probability when E is between V₁ and V₂. The function Prob₂(V₁ $\leq E < V_2$) denotes the probability of the optimal block size mode of the MB, which has the same spatial position in the frame as the MPEG-2 residual MB, is 16x16 when E is between V₁ and V₂. When V₁=0, 100, 200, ... 3900, set V₂=V₁+100. When V₁=4000, set V₂=+ ∞ .

Figure 2 depicts the average value of functions $Prob_1$ and $Prob_2$ for the eleven training video sequences.



Fig.2. Statistics of 16x16 mode and MB

Here, all the MB optimal block size modes used to evaluate functions Prob_1 and Prob_2 are obtained by using exhaustively full search algorithm in advance from eleven real MPEG-2 video sequences, namely, "fore-man", "stefan", "coastguard", "mother & daughter" and "news" with 300 image frames respectively, "silent", "dancer", "kiel", "singer" and "template" with 240 image frames respectively; "flower" with 180 image frames. Each

video sequence represents a different class of motion. In fact, these eleven video sequences can be viewed as the training video sequence.

From the Figure 2, we have the following observations: The smaller of the energy of residual MB, the higher of the probability of the optimal block size mode of the corresponding MB is 16x16 mode. From our experimental results, we get that the $Prob_1(E < 300)=65.2\%$, $Prob_2(E < 300)=78.4\%$, $Prob_1(E > 1000) = 9.5\%$, $Prob_2(E > 1000) = 5.1\%$. Therefore, we can say that there are 65.2% MBs whose associated MPEG-2 residual energy is lower than 300 and whose probability of optimal block size mode is 16x16 is 78.4\%, and 9.5% MBs whose associated MPEG-2 residual energy is bigger than 1000 and whose probability of optimal size is not 16x16 is 1-5.1%=94.9% on the average.

Intuitively, for a MB, which contains more than one object and these objects may not move in the same direction, using only one motion vector may cause only part of the MB can have good motion compensation and the overall resulting residual energy can be large or the DCT coefficients may be distributed unbalanced due to the mismatch in the remaining part of the MB. For an area with uniform motion or texture, it is better to use a larger block in the motion search. On the other hand, for an area with complex motion or texture, keeping the search results of smaller blocks will be better. We can use the energy of residual MB to represent the complexity of texture of a MB. That is if a MB with significant textures, we should search more block size modes.

3. PROPOSED BLOCK SIZE MODE SELECTION ALGORITHM

Motivated by the above observations and analysis, we treat the saving of computation for VSBM in the view of compression. If the energy of MPEG-2 residual MB is very small, we can turn off the matching process from other block size modes since the performance of 16x16 mode is "good enough" and other mode require more motion vectors. On the other hand, if the energy of MPEG-2 residue is very large, we should directly split the MB into four 8x8 blocks to achieve better performance. If the early termination is not successful, we use the distribution of energy of four 8x8 block of a MPEG-2 residual MB to determine the final mode. ME is only performed within a particular block size mode if that mode is checked according to our scheme.

In addition, we only focus on 16x16, 16x8, 8x16, and 8x8 block size modes decision in this paper. When 8x8 mode is chosen, the exhaustively full search method is used to determine which one of the 8x8, 8x4, 4x8, and 4x4 block size mode should be used.

The proposed algorithm is as follows:

1. Initialization

Define several variables as: LowT : threshold for 16x16 block type. UpperT : threshold for 8x8 block type. SumEng: accumulated energy of inter-predicted MPEG-2 residual MB N: accumulated number of MB used inter-code in MPEG-2 bit-stream.

Set LowT=300, UpperT=1000 and SumEng=0.

Visit each MB whose corresponding MPEG-2 MB is inter-coded and update the variable N by:

$$N=N+1.$$

Then perform the following steps:

Early termination Calculate the energy of corresponding MPEG-2 residual MB of current MB E_{16x16} as defined above.

If E_{16x16} < LowT, choose 16x16 as final block type. Else if E_{16x16} > UpperT, choose 8x8 as final block type. Else go to the next step.

3. Block segmentation

2.

Divide a 16x16 MB into four 8x8 blocks as shown in Figure 3. For each 8x8 block, use the variable array $E_{8x8}[i], i=0,1,2,3$ to represent their energy respectively.



Fig.3. Division of 16x16 MB Define several variables as:

- MinBlock, MaxBlock: the minimum and maximum block energy among the four nonzero 8x8 block in the current residual MB respectively.
- AvgBlock: the average energy of the nonzero 8x8 blocks.

K: an empiric constant. In our experiments, it is set to 0.8

TopMB: the top part 8x8 block energy of MB, equal to $E_{8x8}[0] + E_{8x8}[1]$.

LowMB: the low part 8x8 block energy of MB, equal to $E_{8x8}[2] + E_{8x8}[3]$.

LeftMB: the left part 8x8 block energy of MB, equal to $E_{8x8}[0] + E_{8x8}\,[2].$

RightMB: the right part 8x8 block energy of MB, equal to $E_{8x8}[1] + E_{8x8}[3]$.

Based on the number of $E_{8x8}[i] = 0$, there are three cases needed to considered.

Case 1: Only one 8x8 block energy equal to zero.

Choose the 16x16 mode as the final mode. Because we think the overall prediction result is "good enough".

Case 2: Only two or three 8x8 block energy equal to zero.

Detect if only part of the MB is good motion compensated:

If $((MaxBlock - MinBloc) \leq AvgBlock \times K)$, choose 16x16 as final block type. Else choose 8x8 as final block type.

Case 3: Non 8x8 block energy equal to zero.

Detect the distribution of residual energy:

If $(Abs(TopMB - LowMB) \le 2 \times AvgBlock \times K)$, choose the 16x8 mode as the final mode. Else if $(Abs(LeftMB - RightMB) \le 2 \times AvgBlock \times K)$, choose 8x16 mode as the final mode. Else if $((MasBlock - MinBlock) \le AvgBlock \times K)$, choose 16x16 as the final block type. Else choose 8x8 as the final block type.

4. Threshold update

Threshold value should be adaptive with different scenes or sequences. In this paper, we update the two threshold: LowT and UpperT by following equations.

 $SumEng = SumEng + E_{16x16}$

LowT=Min((SumEng \times 1.5) / N,300)

UpperT=Max((SumEng \times 3) / N,1000)

The const 1.5 and 3 is empiric constants, and 300 and 1000 is come from the above observation in our extensive simulation.

4. SIMULATION RESULTS

The results of the proposed algorithm are compared to two CPDT architecture transcoders, one use the full search for all block-sizes and another use only the 8x8 block-size for ME. Simulations are performed on six standard video sequences in CIF (352x288) format. The 300 frames of each of the six sequences are first encoded with MPEG-2 at 1.5 Mbps and at a frame rate of 30 fps, then transcoded to H.264 with quantizer values of 28, a search range of 16, 5 references and with RDO concepts in the H.264 re-encoder. The MPEG-2 encoder and the H.264 encoder [4] both use only the frame prediction and frame picture and the "IPPPP..." sequence (include only I and P frames, no B frames) is used with a GOP size of 15 frames. The software was tested on a computer based on an Intel Pentium IV 1.8GHz CPU with 256 MB RAM and Windows 2000 professional operating system.

Figure 4 shows the frame-by-frame PSNR comparisons for the first 90 frames of the Coastguard sequence.



Fig.4. Frame-by-frame PSNR comparisons for the Coastguard sequence

The results are shown in Table 1 on the respect of PSNR (in dB), bit rates and transcoding time requirement (in terms of microseconds used), respectively. The relative improvements are shown inside the parenthesis in the respective column for the ease of comparison. Compared with the CPDT architecture transcoder with the exhaustively full search algorithm of all the seven block

sizes in H.264, our proposed algorithm can reduce computational cost up to 22% with negligibly small PSNR degradation (0.12dB) and slight increase in bit rate (2.9%) on the average. The CPDT transcoder with only 8x8 block size mode enabled can reduce computational cost up to 24.7% with PSNR degradation (0.15dB) and increase in bit rate (5.6%) on the average. Although getting almost exactly the same speed up in transcoding time for the same reduction in objective quality, our proposed algorithm outperforms the CPDT transcoder with only 8x8 block size mode enabled by appreciable gain on bit rate saving (2.7%).

5. CONCLUSION

In this paper, we propose new algorithm focusing on 16x16, 16x8, 8x16 and 8x8 (which enable 8x8, 8x4, 4x8, 4x4 block types at the same time in our paper) to alleviate the complexity of VSBM in MPEG-2 to H.264 transcoding while maintain the similar visual quality and bit rate. Instead of searching through all possible block type, the proposed method tries to predict the best block types.

Simulation results showed that our method can save 22 % of the whole transcoding time on the average while keeping the

quality nearly the same as the CPDT transcoder with exhaustively full search scheme.

6. ACKNOWLEDGEMENT

This work is supported by National Nature Science Foundation of China under grant number 60302028.

7. REFERENCES

[1] ITU-T Rec: H.264/ISO/IEC 11496-10:'Advanced video coding', Final Committee Draft, Document JVT-G050, December 2002

[2] T.Wiegand. G.Sulivan. G.Bjontegaard, and A. Luthra. "Overview of the H.264/AVC Video Coding Standard." IEEE Transactions on Circuits and Systems for Video Technology, Vol.13, No.7, July 2003

[3] Hari Kalva. "Issues in H.264/MPEG-2 Video Transcoding". Consumer Communications and Networking Conference, 2004. CCNC 2004. First IEEE, 5-8 Jan. 2004

[4] JVT Reference Software official version jm 7.2 http://bs.hhi.de/%7Esuehring/tml/download/jm72.zip

	Coastguar	ď		
	Complexity	PSNR	Bit-rate	
	(ms)	(dB)	(kbit/s)	
CPDT(Full)	7513976	34.96	1204.7	
CPDT(8x8)	5782464	34.76	1250.8	
(saved)	(23.0%)	(-0.20)	(-3.8%)	
Proposed	6067396	34.82	1232.2	
(saved)	(19.3%)	(-0.14)	(-2.3%)	
Foreman				
	Complexity	PSNR	Bit-rate	
	(ms)	(dB)	(kbit/s)	
CPDT(Full)	7036771	36.55	668.4	
CPDT(8x8)	5282289	36.39	714.5	
(saved)	(24.9%)	(-0.16)	(-6.9%)	
Proposed	5482992	36.42	688.5	
(saved)	(22.0%)	(-0.13)	(-3.0%)	
Mother & Daughter				
	Complexity	PSNR	Bit-rate	
	(ms)	(dB)	(kbit/s)	
CPDT(Full)	6401929	38.79	186.4	
CPDT(8x8)	4813302	38.63	204.1	
(saved)	(24.8%)	(-0.16)	(-9.5%)	
Proposed	4888910	38.67	194.7	
(saved)	(23.6%)	(-0.12)	(-4.5%)	

Table.1. Comparison of PSNR, Bit-rate and complexity for two CPDT transcoders and transcoder with proposed algorithm

News				
	Complexity	PSNR	Bit-rate	
	(ms)	(dB)	(kbit/s)	
CPDT(Full)	6942234	37.75	340.1	
CPDT(8x8)	5207573	37.59	354.9	
(saved)	(25.0%)	(-0.16)	(-4.4%)	
Proposed	5283565	36.60	348.4	
(saved)	(23.9%)	(-0.15)	(-2.4%)	
Silent				
	Complexity	PSNR	Bit-rate	
	(ms)	(dB)	(kbit/s)	
CPDT(Full)	7246640	35.39	396.1	
CPDT(8x8)	5377732	35.29(-	417.3	
(saved)	(25.8%)	0.10)	(-5.4%)	
Proposed	5510224	35.27	407.0	
(saved)	(24.0%)	(-0.12)	(-2.8%)	
Stefan				
	Complexity	PSNR	Bit-rate	
	(ms)	(dB)	(kbit/s)	
CPDT(Full)	8075287	35.46	2046.5	
CPDT(8x8)	6066628	35.32	2115.7	
(saved)	(24.9%)	(-0.14)	(-3.4%)	
Proposed	6524464	35.39	2093.3	
(saved)	(19.2%)	(-0.07)	(-2.3%)	