A Fast Watermarking System for H.264/AVC Video

Cong-Van Nguyen, David B. H. Tay, Guang Deng
Department of Electronic Engineering
La Trobe University
Bundoora, VIC3086, Australia
Emails: {c11nguyen@students., D.Tay@, D.Deng@}latrobe.edu.au

Abstract— **In this paper, we propose a fast watermarking system that works on the H.264/AVC motion vectors. By restricting access to DCT coefficients and pixel information, the computational complexity of the watermark embedder/extractor is kept low and much lower than that of the H.264 decoder. The error propagation due to motion prediction compensation is monitored and its effect is limited by a tracking method that is based solely on the motion information from the bitstream. Although this work focuses on the H.264/AVC standard, the novel watermarking technique is also applicable to the MPEG1-2 and MPEG4 video standards.**

Keywords—**watermarking, H.264/AVC, motion vector**

I. INTRODUCTION

Watermarking video signal developed from the well-known spread-spectrum technique of Hartung and Girod [1]. Generally, all proposed video watermarking systems face the same challenge; namely the computational burden due to the massive data rate of typical video signal. Very high MIPS processing platforms are therefore required, but this is not feasible in certain applications. Watermarking directly in the compressed domain is an obvious solution to the computational complexity problem. The coded video data rate is usually hundreds time smaller than the raw data rate. This approach therefore helps to save machine cycles or hardware gates significantly, thus saving power. This is also practical because video data is usually transmitted and exchanged in a compressed form.

Some watermarking systems have been proposed in the literature for use with well known video compression standards (MPEG1-2, MPEG4, and H.264/AVC). In [1], the DCT coefficients of MPEG2 video are modified by using a spread spectrum scheme to carry the watermark bits. On the other hand, the system in [6] embeds the watermark bits in the LSB bits of the motion vectors. Up till now, most of the compressed-domain video watermarking systems rely on marking either the DCT coefficients or the motion vectors. Although watermarking DCT coefficients has the advantage of being more robust to distortion/attacks, watermarking motion vectors is much simpler in terms of computational complexity. In newer video compression techniques (MPEG4, H.264), the encoding of the DCT coefficients into the bitstream format is much complicated than that in the older ones (MPEG1, MPEG2), so the access to DCT coefficients requires more machine cycles.

Watermarking either the DCT coefficients or the motion vectors changes the value of some pixels in a watermarked frame. This distortion is then propagated to the succeeding frames due to motion prediction compensation, even though these frames may not be watermarked at all. In [1][5], drift compensation was used to prevent this error propagation, but this method required a partial reconstruction of some pixels (motion compensation prediction is done on pixel data but not DCT coefficients). Some other watermarking systems [3][4][6] avoid using drift compensation by attaching the watermark embedder with the video encoder. This approach however increases the computation burden significantly.

In this paper, we introduced a different approach for watermarking H.264 motion vectors. Error propagation is limited in our method by restricting access to the DCT coefficients. A scheme of motion vector selection to limit the error propagation is proposed. In addition, we will describe the watermark embedding and extracting schemes in detail. We will also provide some timing results to argue that our watermarking system is more computationally efficient than those proposed in the literature.

II. PROPOSED METHOD

A. Watermarking H.264 motion vectors

In our watermarking system, the watermark bits are embedded in the motion vectors of H.264 video. A motion vector $MV = \{MVX, MVY\}$ contains a horizontal component MVX and a vertical component MVY. Each component is an integer number specifying either the horizontal or vertical dimension of the motion vector. One unit of the component length corresponds to ¼ pel (H.264 supports quarter-pel motion prediction). In the current version of our watermarking system, the two least significant bits of the larger component in a marked motion vector contain the watermark information. Before being watermarked, the motion vector component is quantized as following:

$$Q(MVX) = \begin{cases} (2 + MVX) \,\&\, (0xFFFC) & MVX \geq 0 \\ -((2 - MVX) \,\&\, (0xFFFC)) & MVX < 0 \end{cases}$$

where the logical AND operation (&) is used to clear the two LSB bits. This process quantizes the motion vector component

This work was supported in part by Analog Devices Australia Pty. Ltd., unit 3, 97 Lewis Rd., Wantirna, VIC3152, Australia.

to the nearest full-pel position, so the maximum distortion of a marked motion vector will be only ½ pel (in one dimension). The watermark bits are embedded into the larger motion vector component as follows (where $|Q(MVX)| < |Q(MVY)|$ is assumed without loss of generality) :

$$\overline{MVY} = \begin{cases} Q(MVY) - WB & if\ Q(MVY) \geq 0, WB = 2 \\ Q(MVY) + WB & otherwise \end{cases}$$

where WB contains the watermark bits (WB = -1, 0, 1, or 2; corresponding to one of the bit pairs 11, 00, 01, 10). Note that the two original LSB bits are NOT simply replaced by the watermark bits in this scheme. This is to ensure that the distortion of watermarked motion vectors is minimized. This embedding process ensures the synchronization condition:

$$Q(\overline{MVX}) = Q(MVX) \text{ or } Q(\overline{MVY}) = Q(MVY)$$

Only the larger motion vector component is modified:

$$\overline{MV} = \begin{cases} \{\overline{MVX}, MVY\} & if\ |Q(MVX)| \geq |Q(MVY)| \\ \{MVX, \overline{MVY}\} & if\ |Q(MVX)| < |Q(MVY)| \end{cases}$$

In the H.264/AVC standard, motion vectors are differentially coded as: $MV = MVP + MVD$. MVP is the motion vector prediction, which is derived from the neighboring motion vectors MV_A (left), MV_B (top), and MV_C (top-right or top-left). MVP could be equal to MV_A, MV_B, MV_C, or *median*(MV_A, MV_B, MV_C) depending on the macroblock mode and the availability of neighboring blocks). MVD is the motion vector difference, which is encoded into the H.264 bitstream using Exp-Golomb code (Baseline profile). Thus, the required changes in the watermarked motion vectors should be converted into the required changes in the motion vector differences. The changes are then applied to the motion vector differences before being re-encoded into the watermarked bitstream. The important point here is that changing one motion vector requires changing not only the corresponding motion vector difference but also those of the neighboring motion vectors to ensure that all motion vectors will be restored in the watermark extractor/decoder to their correct values. This process is called motion vector error compensation, and it may contribute to an increase in the bitstream size.

B. *Selecting motion vectors for watermarking*

We found that watermarking very small motion vectors causes blocks in slow-motion areas to be distorted and these distortions are very noticeable. In the proposed system, motion vectors whose length is less than 2 (½ pel) will not be watermarked at all.

Another restriction comes from the motion vector prediction mechanism. The motion vector of a skipped macroblock is derived from only MVP (there is no MVD data for a skipped macroblock in the bitstream). Therefore, a motion vector corresponding to a skipped macroblock can not be watermarked due to the lack of the MVD. Moreover, if the neighbor affecting this motion vector is watermarked, the motion vector error in the skipped macroblock cannot be compensated. Therefore, the neighbor(s) of a skipped macroblock should be unmarkable as well. (Actually, we could watermark skipped macroblocks by converting the skip mode into a normal mode, but this would cause the VLC code size to increase significantly).

The motion vectors remaining after the above restrictions are watermarkable. We watermark these vectors selectively to limit the watermark distortion. The *least-referenced blocks* are chosen for watermarking based on the *referenced cost* criterion.

C. *Sorting motion vector*

Let us denote the i-th video frame in a H.264 video sequence by F_i . Each frame is M macroblocks in width and N macroblocks in height (the frame contains $16M$x$16N$ pixels). The j-th macroblock in F_i is denoted by MB_{ij} . Each macroblock is divided into variable-size blocks (16x16, 16x8, 8x16, 8x8, 8x4, 4x8, or 4x4 pixels). The k-th block in MB_{ij} is denoted by B_{ijk} . B_{ijk} is associated with a reference index REF_{ijk} and a motion vector $MV_{ijk} = \{MVX_{ijk}, MVY_{ijk}\}$. The predicted pixels of the block B_{ijk} is derived from the reference block in a reference frame indexed by REF_{ijk} (H.264/AVC allows multiple reference frames). The reference block is denoted by $RB(B_{ijk}, MV_{ijk}, REF_{ijk})$.

The *referenced cost* $RC(B_{ijk})$ of B_{ijk} is computed as:

$$RC(B_{ijk}) = \sum_a \sum_b \sum_c \text{overlap}(B_{ijk}, B_{abc})$$

where a is the index of the succeeding frames which may reference to F_i , b is the index of the macroblocks in F_a , c is the index of blocks in MB_{ab} , overlap(B_{ijk},B_{abc}) is the number of common pixels between $RB(B_{ijk},\{Q(MVX_{ijk}),Q(MVY_{ijk})\}, REF_{ijk})$ and B_{abc} (the quantized motion vector components are used to avoid fractional-pel interpolation and ensure the synchronization between the embedder and the extractor).

The sorting cost for each block in the current frame is computed as following:

$$SC(B_{ijk}) = \text{area}(B_{ijk}) + RC(B_{ijk})$$

where area(B_{ijk}) is the number of pixels in B_{ijk} . Visually, $SC(B_{ijk})$ specifies the number of distorted pixels which will be introduced to the current frame and the succeeding frames if B_{ijk} is watermarked. Within the current frame, those motion vectors having smallest sorting costs will be chosen for watermarking.

III. WATERMARK EMBEDDING

Our watermark embedder works directly on the H.264 bitstream syntax elements. It locates the motion vector differences (MVD) and modifies the corresponding VLC representation (Exp-Golomb code) to produce the watermarked bitstream. Because processing data from the Exp-Golomb code is simpler than from other entropy codes (Context-Adaptive Variable Length Coding CAVLC, Context-Adaptive Binary Arithmetic Coding CABAC), watermarking motion vectors would be the fastest method for watermarking H.264 video.

Although the embedder needs only the position of the MVDs in the cover bitstream, it should parse all syntax elements that are present. This means decoding the whole VLC layer is required, but irrelevant decoded data (luma and chroma

coefficients, intra prediction information) could be omitted to save memory. Some important data such as parameter sets, slice headers, macroblock headers, and sub-macroblock headers must be decoded accurately however.

The following **pre-watermarking process** is applied to the currently decoded frame F_i :

- If F_i is an I frame, the embedder will proceed to the next frame by omitting all NAL units belonging to F_i. Otherwise, the steps below will be performed (F_i is then a P frame).
- All motion vectors in F_i are decoded, and the positions of the corresponding MVDs in the bitstream are stored.
- The *sorting cost* of each motion vector is initialized with the area (in pixels) of the corresponding block.
- For each motion vector MV_{ijk} , the reference frame F_x is located by using REF_{ijk} . For each pixel in B_{ijk} , the embedder finds the referenced pixel in F_x (using the quantized motion vector $\{Q(MVX_{ijk}),Q(MVX_{ijk})\}$) and the block B_{xyz} containing this referenced pixel, and then adds 1 to the sorting cost of MV_{xyz} (if the quantized motion vector points to the outside of the reference frame, the referenced pixel will be the nearest pixel on the frame edges).
- Store F_i into the decoded picture buffer (DPB) and proceed to the next frame.

When the frame F_i is removed from DPB (or F_i is no longer referenced), the **watermarking process** is performed as following:

- In F_i , all motion vectors shorter than 2 (½ pel) and all motion vectors belonging to skipped macroblocks and their neighboring blocks are labeled as unmarkable.
- If the number of watermarkable motion vectors in F_i is less than the *watermarking rate* L (in motion vectors per frame MV/f), F_i will not be watermarked at all and the steps below will be skipped.
- All watermarkable motion vectors in F_i are sorted. The L motion vectors having the smallest sorting cost are chosen for watermarking using the following steps.
- Embed watermark bits into the chosen motion vectors as described in section 2.1.
- Modify the MVDs corresponding to the watermarked motion vectors to realize the above embedding process.
- Modify the MVDs corresponding to the neighbors of watermarked motion vectors to prevent error propagation due to motion vector prediction.
- Repack all changed MVDs into the output bitstream.

There is frame latency between the being watermarked frame and the being decoded frame. H.264 standard supports at maximum 16 reference frames at a time, so the longest latency is 16 frames. In general, long-term reference frame should not be watermarked to avoid longer latency.

IV. WATERMARKING EXTRACTION

The watermark extractor is very similar to the embedder. It contains the following common processes: decoding motion vectors, calculating sorting cost for each motion vectors, sorting and selecting watermarked motion vectors. The watermark bits embedded in each motion vectors are simply extracted from the two LSB bits of the longer motion vector component. We implement two kinds of watermark extractor: *stand-alone extractor* and *decoder-accompanied extractor*. The stand-alone extractor only reads the watermark bits from the watermarked bitstream while the decoder-accompanied extractor also performs decompressing the video.

V. SIMULATION RESULTS

We implemented the watermark embedder and the watermark extractor based on JM decoder (version 9.6). Some unnecessary parts of the decoder such as intra/inter prediction compensation, fractional-pel interpolation, dequantizing, and DCT transform are de-activated to reduce the computational complexity. Currently, our watermarking system supports only the H.264 Baseline profile.

Some well-known video sequences (Akiyo, Foreman, Flower…) in raw format (YUV 4:2:0) are compressed using the JM encoder in Baseline profile to produce some test H.264 bitstreams for our system. Fig. 1 and Fig. 2 show the last frame of Foreman bitstream in unwatermarked form and watermarked form. The simulation results with test bitstreams are shown in Table I (we used an all-0-bit watermark message for all test bitstreams without any lost of generality). The PSNR is computed by comparing the watermarked video and the cover video (after being decompressed). The PSNR is somewhat dependent on the nature of each specific video.

The watermarking efficiency rate varies widely. For the bitstream Container QCIF, 183 extra bitstream bytes are required to embed only 159 watermark bytes; but for the bitstream Flower QCIF, embedding 2975 watermark bytes needs only 975 extra bytes. This is due to a natural feature of the exponent Golomb code: changing small value codes (near 0) is more likely to change the code length than changing large value codes. In the Container bitstream, the motion is low and homogeneous, so most MVDs are very small. On the other hand, the motion in the Flower bitstream is high and non-homogeneous (with zooming), so there are more large-value MVDs. Nevertheless, the amount of overhead data seems to be much smaller than the size of the whole bitstream because we cannot watermark too much motion vectors in a frame (unless high distortion is tolerable).

The computational complexity of the watermark embedder as well as the stand-alone watermark extractor is much lower than that of the full H.264 decoder. For some high-compression-rate bitstreams such as Akiyo and Container, the computation time of the stand-alone watermark extraction is only 20% of the time needed for the general decoder. The decoder-accompanied watermark extractor is also efficient; it spends up to about 112% of the decoder time to do both watermark extraction and video decompression.

TABLE I. SIMULATION RESULTS FOR DIFFERENT VIDEO SEQUENCES

Video Sequences	PSNR		Payload	Bitstream Size (bytes)			Decode	Embed		Extract		Decode+Extract	
	Avg	Min	(bytes)	Org	Marked	Diff	Time (s)	Time (s)	Saving	Time (s)	Saving	Time (s)	Excess
Akiyo CIF @ 20MV/f	41.33	37.98	495	45194	45576	382	11.276	1.752	84.46%	1.673	85.16%	11.617	3.02%
Bus CIF @ 100MV/f	37.71	35.15	2975	572202	573358	1156	18.807	5.368	71.46%	5.257	72.05%	19.759	5.06%
Coastguard CIF @ 100MV/f	37.27	32.65	2975	648038	649437	1399	19.989	6.309	68.44%	6.159	69.19%	20.950	4.81%
Container CIF @ 10MV/f	38.65	35.20	295	89957	90268	311	12.177	2.043	83.22%	2.023	83.39%	12.568	3.21%
Flower CIF @ 100MV/f	34.14	32.11	2975	804772	805747	975	18.697	6.890	63.15%	6.710	64.11%	20.019	7.07%
Foreman CIF @ 100MV/f	36.96	32.24	2975	180550	181986	1436	15.021	2.834	81.13%	2.825	81.19%	15.663	4.27%
Mobile CIF @ 100MV/f	33.31	30.86	2975	883651	885035	1384	21.761	8.021	63.14%	7.811	64.11%	23.404	7.55%
Tempete CIF @ 100MV/f	34.81	32.42	2975	548487	550423	1936	18.817	5.578	70.36%	5.448	71.05%	20.089	6.76%
Waterfall CIF @ 40MV/f	36.51	31.58	1190	150843	152001	1158	14.982	2.724	81.82%	2.634	82.42%	15.573	3.94%
Akiyo QCIF @ 8MV/f	38.79	36.00	196	20236	20402	166	4.727	0.901	80.94%	0.902	80.92%	4.958	4.89%
Carphone QCIF @ 24MV/f	34.72	31.72	1116	90335	91050	715	6.339	1.442	77.25%	1.422	77.57%	6.579	3.79%
Coastguard QCIF @ 24MV/f	36.21	31.68	1194	189913	190472	559	7.591	2.213	70.85%	2.193	71.11%	7.872	3.70%
Container QCIF @ 4MV/f	29.35	26.31	159	31157	31340	183	4.977	1.001	79.89%	0.992	80.07%	5.167	3.82%
Foreman QCIF @ 32MV/f	33.52	31.64	1552	102867	103697	830	6.720	1.542	77.05%	1.532	77.20%	7.000	4.17%
Mobile QCIF @ 24MV/f	31.23	28.10	1194	358210	359271	1061	9.163	3.314	63.83%	3.245	64.59%	9.594	4.70%
Salesman QCIF @ 8MV/f	40.77	36.67	330	42984	43253	269	5.088	1.092	78.54%	1.071	78.95%	5.678	11.60%

PSNR is calculated by comparing the watermarked sequence and the unmarked sequence (only luma component)

Each QCIF sequence contains 200 frames, each CIF sequence contains 120 frames

Encoding parameters: Baseline profile, QP=28, 3 reference frames, SearchRange=16 (for QCIF) or 32 (for CIF)

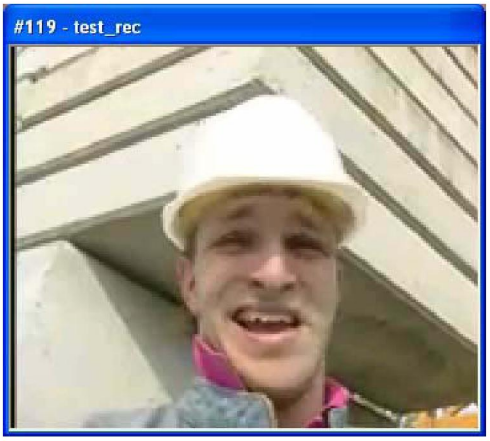

Figure 1. Unwatermarked video frame

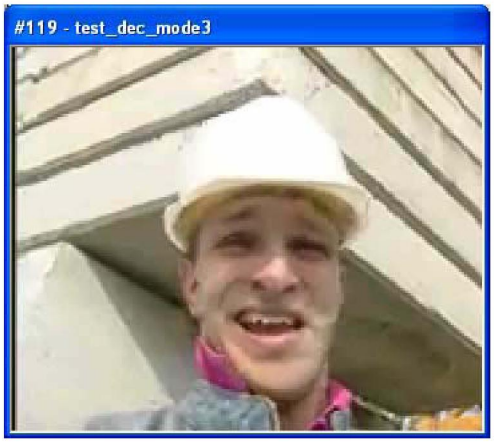

Figure 2. Watermarked video frame

VI. CONCLUSION

A new method for watermarking H.264/AVC motion vector video has been proposed. Its significant improvement in comparison with the previous systems in the literature is that both the watermark embedder and watermark extractor are much simpler than the full video decoder (one third or less the complexity). Only the motion vector difference information is changed in the embedding process, so the structure and the size of H.264 bitstream is not changed significantly.

ACKNOWLEDGMENT

The authors would like to thank Mr. Robert Slaviero, Mr. Simon Brewer, and Mr. Stephen To of Analog Devices Australia Pty. Ltd. for supporting the work reported here.

REFERENCES

[1] F. Hartung, B. Girod, "Digital watermarking of MPEG-2 coded video in the bitstream domain", *the 1997 IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 4, pp. 2621-2624, Apr. 1997.

[2] J. Zhang, J. Li, and L. Zhang, "Video watermark technique in motion vector", *Proceedings of the XIV Brazilian Symposium on Computer Graphics and Image Processing*, pp. 179-182, 2001.

[3] Z.-J. Zhu, G.-J. Jiang, M. Yu, X.-W. Wu, "New algorithm for video watermarking", *Proceedings of the 2002 International Confererence on Signal Processing ICSP'02*, vol. 1, pp. 760-763, Aug 2002.

[4] Z. Zhao, N.-H. Yu, X.-L. Li, "A novel video watermarking scheme in compressed domain based on fast motion estimation", *Proceedings of the 2003 International Conference on Communication Technology ICCT2003*, vol. 2, pp. 1878-1882, Apr 2003.

[5] A. M. Alattar, E. T. Lin, and M. U. Celik, "Digital Watermarking of Low Bit-Rate Advanced Simple Profile MPEG-4 Compressed Video", *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 8, pp. 787-800, Aug. 2003.

[6] G. Qiu, P. Marziliano, A. T. S. Ho, D. He, and Q. Sun, "A hybrid watermarking scheme for H.264/AVC video", *Proceedings of the 17th International Conference on Pattern Recognition (ICPR'04)*, vol. 4, pp. 865-868, Aug 2004.