A NOVEL APPROACH TO FAST MULTI-FRAME SELECTION FOR H.264 VIDEO CODING

Andy Chang, Oscar C. Au, Y. M. Yeung
Department of Electrical and Electronic Engineering
The Hong Kong University of Science and Technology
Email: eecax@ust.hk, eeau@ust.hk, eeyym@ust.hk

ABSTRACT

The latest under development video coding standard, H.264, uses multiple reference frames to improve the rate-distortion performance. However, the motion estimation process involved is computational intensive and increases linearly with the number of allowed reference frames. In this paper, a novel fast multi-frame selection method is proposed for H.264 video coding. The proposed scheme can efficiently reduce the computational cost up to 70% with similar quality and bit-rate. So the method is highly suitable for real-time (or low delay) applications while the benefits from multi-frame motion compensation can be preserved.

1. INTRODUCTION

The uses of motion estimation/motion compensation (ME/MC) in video coding significantly lower the bit-rate in video compression with the use of more memory and computational power. With the continue development of semiconductor industry, we are now able to increase the memory in video coder such that more frames can be stored for ME/MC. This can further reduce the temporal redundancy in video sequences.

The H.264 is a developing international video coding standard. It can provide both objective and subjective image quality superior to existing standards. The basic encoding algorithm of H.264 [1] is similar to H.263 or MPEG standard except integer 4x4 discrete cosine transform (DCT) is used instead of the traditional 8x8 DCT. Additional feature including intra prediction mode for I-frames, seven block sizes for motion estimation, multiple reference picture selection are used in H.264 for higher coding efficiency.

The multiple reference picture selection in H.264 allows the encoder store up to five previous frames for ME/MC. However, the processing time increases linearly with the number of allowed reference frames. This is because motion estimation is performed for each reference frame. This full selection process provides the best coding result but the computational cost is very high.

In this paper, we propose a novel fast multi-frame selection scheme that can efficiently reduce the computational cost while keeping the similar quality and bit-rate. Instead of searching through all the possible reference frames, the proposed scheme try to select some representative reference frames for motion estimation. The proposed method can obtain an image quality and bit-rate close to the full selection process while the process time can be greatly reduced. This is very useful for real-time applications.

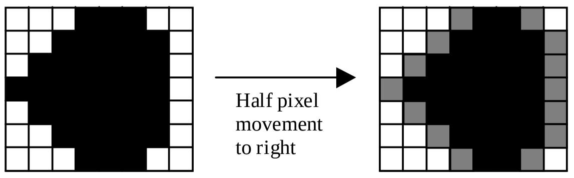

Fig. 1a. Example of Half pixel movement to the right.

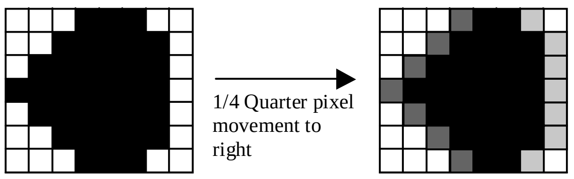

Fig. 1b. Example of 1/4 Quarter pixel movement to the right.

2. OBSERVATIONS ON MULTI-FRAME MOTION ESTIMATION

A digital video sequence is a discrete representation of the continuous changing world. An image pixel value is the integration of the light intensity from some objects that falls into the detection range of some discrete detector like a CCD sensor. Between consecutive video frames, the objects may move leading to the possibility of using motion estimation and compensation to reduce temporal redundancy. Typically translational motion is assumed. Between two consecutive frames, an object motion may sometimes correspond to an (approximately) integer multiple of the inter-sensor distance leading to integer pixel motion in the video frames (e.g. 3 pixel to the left). Sometimes the object motion is more like a half-pixel motion (e.g. 1.5 pixel to the right). Sometimes the object motion is a quarter-pixel motion (e.g. 3.25 pixel or 5.75 pixel to the right).

The basic idea of sub-pixel motion estimation algorithm, like half-pixel or quarter-pixel estimation, uses interpolation to predict the sub-pixel shift of texture relative to the sampling grid. An accurate prediction can efficiently reduce the degree of error between the original image and the predicted image. This is useful for background texture where panning motion happens frequently. For this reason, quarter pixel accuracy motion estimation is adopted in H.264 for better compression performance.

Suppose an object has edges aligned perfectly with the sensor boundaries at a particular time instant such that the object is clear and sharp. We will describe this object as having "integer-pixel location". When the object undergoes an integer-pixel translational motion, the object will look exactly the same in the two consecutive frames except that one is a translation to another. And the moved object can be predicted perfectly by integer-pixel motion estimation.

When the object undergoes a half-pixel motion, the edges may be blurred as shown in the example in Fig. 1a. We will describe this object as having "half-pixel location". The zero pixel wide (sharp) object edge now becomes one pixel wide (blurred). The pixel at the blurred object edges may have only half the intensity of the original object, which can lead to difficulty in motion estimation. In particular, the block on the right in Fig. 1a can be predicted perfectly by the block on the left using half-pixel motion estimation. However, the block on the left cannot be predicted perfectly by the block on the right, even if there is no noise and the motion is an ideal translational motion and there is no change to the object lighting.

Similarly, when the object undergoes a quarter-pixel motion, the edges may be blurred as shown in Fig. 1b. We will describe this object as having "quarter-pixel location". The zero pixel wide (sharp) object edge becomes one-pixel-wide (blurred). The pixels at the blurred edges may have 3/4 or 1/4 of the intensity. Again, the blurred "quarter-pixel location" object can be predicted perfectly from the sharp "integer-pixel location" object using quarter-pixel motion estimation, but not the other way round.

In general, the objects can be classified into sub-pixel location classes, namely "integer-pixel location", "half-pixel location" and "quarter-pixel location". The edge (and probably texture) details in the 3 classes are different.

The advantage of multi-frame motion estimation is that it can further reduce the temporal redundancy in the video sequences by considering more than one reference frames. The best match is found by minimizing the cost function:

$$J(\mathbf{m}, \lambda_{MOTION}) = SAD(s, c(\mathbf{m})) + \lambda_{MOTION} \cdot R(\mathbf{m} - \mathbf{p}) \quad (1)$$

with $\mathbf{m} = (m_x, m_y)^T$ being the motion vector, $\mathbf{p} = (p_x, p_y)^T$ being the prediction for the motion vector and λ_{MOTION} being the Lagrange multiplier. The term $R(\mathbf{m} - \mathbf{p})$ represents the bits used to encode the motion information only and are obtained by table-lookup. The SAD (Sum of Absolute Differences) is computed as:

$$SAD(s,\ c(\mathbf{m})) = \sum_{x=1,\ y=1}^{B,\ B} \left| s[x, y] - c[x - m_x, y - m_y] \right| \quad (2)$$

where B = 16, 8 or 4 and s being the original video signal and c being the coded video signal.

There are basically two types of temporal redundancy that can be captured by using multi-frame but not traditional single frame ME/MC. The first type of redundancy is related to short-term memory. Suppose the current frame is frame t. Sometimes, some objects may be distorted or absent in frame t-1, but well represented in frames t-2 to t-5. An example is the blinking of an eye, which is a very fast motion. If multiple reference frames are allowed, the motion estimation and compensation can be significantly better than just one reference frame.

The second type is the sub-pixel movement of textures. As mentioned before, because of the discrete nature of the capturing devices, it is not possible to capture the sub-pixel displacement of texture perfectly. As a result, textures and objects with different version of sub-pixel movement ("integer-pixel location", "half-pixel location" and "quarter-pixel location") may occur in successive video frames. The more previous (reference) frames stored and considered in the buffer, the higher the probability for the current image to find a reference frame with the same "pixel location". It can be seen that multi-frame motion estimation combined with sub-pixel accuracy motion estimation is a strong combination to tackle the sub-pixel movement of texture and edges. It was suggested in [2] that sub-pixel motion compensation becomes less important when multi-frame motion compensation prediction (MCP) is used.

3. PROPOSED FAST SELECTION SCHEME

In our experiments, we observe that there is a great tendency for the cost function to be especially small when the same shifted version of texture is used to do the motion estimation and compensation. And the use of multiple reference frames provides a good probability of finding the same shifted version of texture in the frame buffer for the current macroblock. In other words, the optimal reference frame tends to be the one with the same "sub-pixel location" as the current frame. However, it is difficult to determine accurately the current "pixel location" class *a priori*.

We also observe that when there is more than one frame with the same "sub-pixel location", the one closer to the current frame is usually better. As a result, the motion estimation speed can be increased by estimating the type of shifted version of texture ("sub-pixel location") in the frame buffer. For example in Fig.2, the black square means collocated macroblocks in Frame t and the multiple reference frames. So if the macroblock at Frame t-1 and Frame t-2 show the same shifting characteristic then we can drop the redundant one (Frame t-2) and reduce motion estimation complexity.

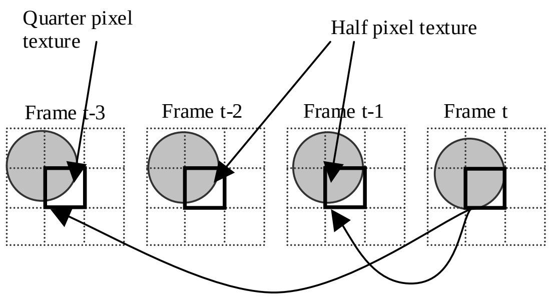

Fig. 2. Selection of Frame for motion estimation

In the proposed multi-frame selection process, each macroblock in each reference frame will have a "sub-pixel location" which is calculated and updated by adding the motion vector obtained in the previous search with the previous "sub-pixel location". For example in Fig.2 every macroblock in Frame t-1, Frame t-2 and Frame t-3 will have their own "sub-pixel location". That "sub-pixel position" will indicate whether the macroblock is an integer-pixel, half-pixel or quarter-pixel type of texture. Before performing motion estimation for the current macroblock or the "black" macroblock in Frame t shown in Fig.2, all the "black" macroblock with the same spatial location in Frame t-1, Frame t-2 and Frame t-3 will be checked such that when two or more macroblocks have the same "sub-pixel location", only one frame is enabled for motion estimation. The "sub-pixel location" of two macroblocks are considered the same if both the x and y component of "sub-pixel location" are equal. The process will enable the frame with smallest temporal distance (closest) to the current frame. So let us look at the example in Fig.2 again. Because both the black macroblock in Frame t-1 and t-2 have the same "sub-pixel location" as "half-pixel location", Frame t-2 will be skipped for in the motion estimation process.

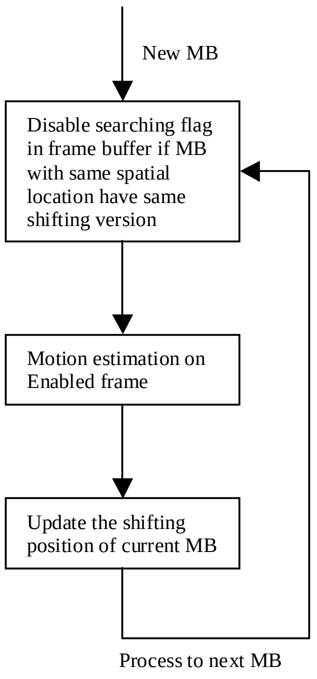
New MB
Disable searching flag in frame buffer if MB with same spatial location have same shifting version
Motion estimation on Enabled frame
Update the shifting position of current MB
Process to next MB

Fig. 3. Flow Chart of the proposed algorithm

In our experiments, the previous reference frame (i.e. frame t-1) has the highest probability of being selected. In our algorithm, motion estimation is always performed in this frame (i.e. frame t-1). After the motion estimation process, the motion vector obtained will be used to update the "sub-pixel location" of the current macroblock. So for Fig.2 if we assume frame t-3 is chose to be the reference frame for the black (current) macroblock in frame t, the "sub-pixel location" of the black macroblock in frame t is obtained by adding the motion vector information to the "sub-pixel position" of black macroblock in frame t-3. The whole process will repeat for the next macroblock. Fig.3 shows the flow chart of the proposed algorithm.

4. EXPERIMENTAL RESULTS

The proposed scheme was tested on four QCIF (176x144) sequences including "Akiyo", "Coastguard", "Stefan" and "Foreman" with QP equals to 16. The proposed scheme was implemented in the H.264 reference software TML9.0 [3]. PMVFAST [4] was implemented in TML9.0 for motion estimation processing.
Table 1 gives the results of PSNR, bitrate and complexity reduction of the above testing sequences using the proposed algorithm. The computational cost is measured by a powerful performance analyzer[1] in term of number of clock cycles used. Compared with the full multi-frame selection scheme in H.264, the proposed scheme can efficiently reduce the computational cost by about 40% of

[1] Inter® Vtune™ Performance Analyzer 6.1

total encoding time. The degradation of the proposed scheme is negligible (about 0.01dB PSNR) with an average increase in bitrate by 0.57%. Fig. 4 shows the PSNR comparison between the optimal scheme in H.264 and the proposed method for the "Coastguard" sequence.

5. CONCLUSION

In this paper, a novel fast multi-frame selection scheme is proposed for H.264 video coding. The proposed scheme skips (does not perform) the motion estimation process on a reference frame if a closer reference frame has the same "sub-pixel location". This scheme can efficiently reduce the computation cost with PSNR quality and bitrate close to the full selection scheme. It's highly suitable for real time applications.

6. ACKNOWLEDGEMENT

This work was supported by Research Grants Council of Hong Kong SAR (HKUST6203/02E).

REFERENCES

[1] Joint Video Team (JVT) of ISO/IEC MPEG and ITU-T VCEG, "Joint Video Specification (ITU-T Rec. H.264 | ISO/IEC 14496-10 AVC) – Joint Committee Draft", document JVT-E022d3.doc, Sep. 2002.

[2] B. Girod, "Efficiency Analysis of Multihypothesis Motion-Compensated Prediction for Video Coding", *IEEE Trans. on Image Processing*, vol. 9, no. 4, Feb 2000.

[3] A. M. Tourapis, O. C. Au, and M. L. Liou, "Predictive Motion Vector Field Adaptive Search Technique (PMVFAST) - Enhancing Block Based Motion Estimation," *in proceedings of Visual Communications and Image Processing 2001 (VCIP-2001)*, pp.883-892, San Jose, CA, January 2001.

[4] JVT Reference Software unofficial version TML9.0, http://bs.hhi.de/~suehring/tml/download/tml90.zip

[5] P. Topiwala, G. Sullivan, "Performance Evaluation of H.26L, TML 8 vs. H.263++ and MPEG-4", document VCEG-N18, Sept. 2001.

[6] P. Topiwala, G. Sullivan, A. Joch, F. Kossentini, "Overview and Performance Evaluation of the ITU-T Draft H.26L Video Coding Standard," *Proc. SPIE, Appl. Of Digital Image Processing*, August 2001.

[7] A. Joch, F. Kossentini, "Performance analysis of H.26L coding features," Document VCEG-O42, VCEG 15th meeting, Pattaya, Thailand, 4-6 December, 2001.

[8] D. Hepper, "Efficiency Analysis and Application of Uncovered Background Prediction in a Low Bit Rate Image Coder", *IEEE Trans. On Communications*, vol. 38, no. 9, Sept. 1990.

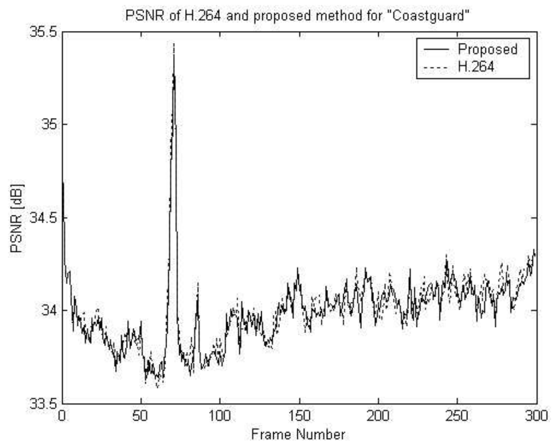

Fig. 4. PSNR of the optimal and proposed method for "Coastguard"

Coastguard			
	Complexity	PSNR(dB)	Bitrate
H.26L	69.6×10^9	34.01	2472144
Proposed	38.5×10^9	34	2479936
Saved	44.7%	-0.01	-0.3%
Akiyo			
	Complexity	PSNR(dB)	Bitrate
H.26L	58.9×10^9	38.23	287336
Proposed	18.01×10^9	38.23	288288
Saved	69.4%	0.00	-0.3%
Stefan			
	Complexity	PSNR(dB)	Bitrate
H.26L	72.6×10^9	34.01	4814040
Proposed	49×10^9	34	4833424
Saved	32.5%	-0.01	-0.4%
Foreman			
	Complexity	PSNR(dB)	Bitrate
H.26L	75×10^9	35.65	1604520
Proposed	39×10^9	35.63	1621200
Saved	34.7%	-0.02	-1.03%

Table 1. Comparison of PSNR, bitrate and complexity for H.264 and proposed algorithm