PRACTICAL REAL-TIME VIDEO CODEC FOR MOBILE DEVICES

Keman Yu[1], *Jiangbo Lv*[*2], *Jiang Li*[1] *and Shipeng Li*[1]

[1]Microsoft Research Asia, Beijing, China
[2]Department of Information and Electronic Engineering, Zhejiang University, Hangzhou, China

ABSTRACT

Real-time software-based video codec has been widely used on PCs with relatively strong computing capability. However, mobile devices, such as Pocket PCs and Handheld PCs, still suffer from weak computational power, short battery lifetime and limited display capability. We developed a practical low-complexity real-time video codec for mobile devices. Several methods that can significantly reduce the computational cost are adopted in this codec and described in this paper, including a predictive algorithm for motion estimation, the integer discrete cosine transform (*IntDCT*), and a DCT/Quantizer bypass technique. A real-time video communication implementation of the proposed codec is also introduced. Experiments show that substantial computation reduction is achieved while the loss in video quality is negligible. The proposed codec is very suitable for scenarios where low-complexity computing is required.

1. INTRODUCTION

In the past years, research on video compression is focused on improving rate-distortion performance. In order to achieve higher coding efficiency, video coding techniques trend to be more elaborate and complex. This leads to higher requirements of the computing capability. More and more users are seeking real-time video communication services with the rapid development of wired and wireless networks. However, real-time software-based video encoding introduces a tradeoff between compression efficiency and complexity. How to reduce the computational requirements as much as possible without obviously compromising the video compression efficiency is one of the key issues in research on video communication.

Recent years have witnessed the rapid development of the mobile devices with wireless connections. With the continuing improvement of storage capacity and computing capability, software-based video coding on mobile devices is becoming economically viable. The emergence of digital cameras for mobile devices also provides conditions for real-time video communication. H.263 [1] is an ITU-T recommended video coding standard and has been broadly used in video conferencing on PCs. However, the conventional H.263 encoder is still computationally expensive and practically not feasible for mobile devices. In our experiments of coding a general video clip in QCIF format, the average frame rate achieved on an iPAQ 3650 Pocket PC is only about 1 fps.

For this reason, we investigated the resource distribution of the H.263 video encoder which uses integer-pixel full search on motion estimation and floating-point DCT. Figure 1 shows the distribution of the execution time for a testing sequence. In this figure, motion estimation, DCT/IDCT and quantization which utilize 73%, 16% and 4% of the execution time respectively are the three primary computationally-intensive components. Thus, the goal of this paper clearly is to improve these three components.

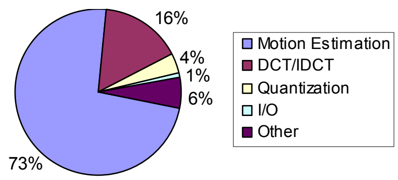

Figure 1: Execution-time distribution for Miss America sequence in QCIF.

The rest of this paper is organized as follows. Section 2 describes the approaches we use to speed up video compression. Experimental results are shown in Section 3. Section 4 presents a practical video communication implementation on mobile devices. Finally, Section 5 concludes this paper and gives future directions.

2. APPROACHES

Figure 2 shows the diagram of the proposed video encoder. The structure is similar to that of the H.263 encoder and is compatible with the H.263 bitstream syntax. The Rate Control component enables the output bit rates to be adaptive to the target. Motion estimation, DCT/IDCT and Quantizer are three critical modules which consume the majority of the execution time in this architecture. The rest of this section will introduce approaches that can speed up the execution of these three modules.

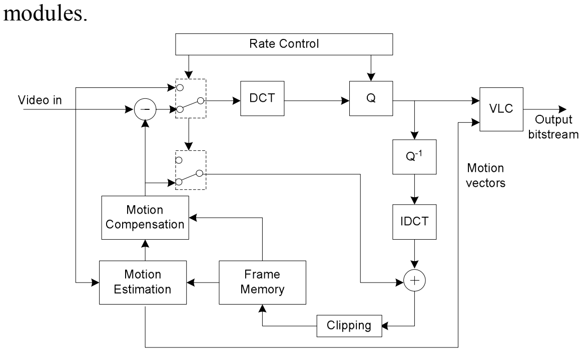

Figure 2: The structure of the proposed video encoder.

[*] The work presented in the paper was carried out at Microsoft Research Asia.

2.1 Low-Complexity Motion Estimation Algorithm

Motion estimation (ME) is efficient in eliminating temporal redundancy between adjacent frames. At the same time, motion estimation is also regarded as a vital component in a video encoder as it consumes the largest amount of computational resources. There are significant advances in fast motion estimation techniques in recent years for alleviating the heavy computation load, such as the diamond search (DS) [2], the small-cross-diamond search (SCSD) [3] and the predictive algorithm (PA) [4]. The complexity of the PA algorithm is the lowest among these fast motion estimation approaches, and its time is relatively constant as there is no recursive searches in this algorithm. We analyze the PA algorithm and propose an appropriate number of candidate predictors and an adaptive motion vector (MV) refinement method.

The algorithm utilizes the characteristics that macroblocks (MB) close to the current MB, in time and in space, are highly probable to have the same motion. Instead of testing all possible MVs as full search (FS) does, the previously calculated MVs of the contiguous MBs comprise a set of candidate predictors, then the best candidate predictor, i.e. the MV with the lowest sum of absolute differences (SAD), is corrected by the refinement phase.

In our investigation, the candidate predictors set which is composed of three MVs plus a null vector, as shown in Figure 3, achieved a good trade-off between computational savings and performance loss.

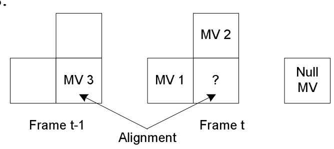

Figure 3: The candidate predictors set: two spatial predictors belonging to the same frame, one temporal predictor belonging to the previous frame and a null vector.

The goal of the refinement phase is to make the best predictor approximate to the real motion. The method of PA is to search the optimal MV at the eight points around the best predictor. An adaptive search approach is proposed in this paper. The SAD of the best predictor is evaluated for choosing the testing points. The refinement processes are divided into the following three cases according to the comparison results of the SAD and two thresholds, TH_1 and TH_2, as shown in Figure 4.

Case 1: SAD <= TH_1, only four points on the cross direction are tested;

Case 2: TH_1 < SAD <= TH_2, eight points around the best predictor are covered;

Case 3: SAD > TH_2, eight points which are two pixels away from the best predictor are involved.

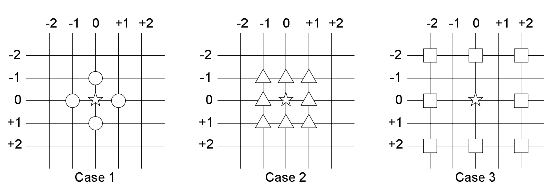

Figure 4: Refinement vectors in three different cases.

In some circumstances, for example when scene changes or there is a sudden motion change, the best predictor is not reliable, which can be indicated by a large SAD value. Increasing the search area, as shown in case 3, addresses this issue. Suitable values are TH_1=4000 and TH_2=6000 according to our experiments on many sample sequences. Table 1 shows the distribution of refinement vectors, which indicates that case 1 covers over 95% of the cases. Consequently, the average number of SAD operations per MB shown in this table is fairly small. Table 2 compares the performance between PA and FS. Using the PA method, a 18.5 to 20-fold speed increase on motion estimation is achieved whilst the average Peak Signal-to-Noise Ratio (PSNR) degradation is marginal.

Table 1: Distribution of refinement vectors and average number of SAD operations per MB for the QCIF sequences coded at 15 fps and 56 kbps

	Case1	Case2	Case3	SAD OP
Miss_am	99.99%	0.01%	0	4.86
News	98.17%	1.40%	0.43%	4.85
Carphone	95.71%	2.91%	1.38%	5.51
Coastguard	96.82%	2.93%	0.25%	6.04

Table 2: Average PSNR degradation and speed improvement ratio of the ME module comparing to FS with a search distance of 15 pixels

	PSNR loss (dB)	Speed improvement ratio
Miss_am	0.13	18.95
News	0.00	18.44
Carphone	0.20	19.31
Coastguard	0.13	19.88

2.2 Integer DCT and inverse DCT

The discrete cosine transform (DCT) is widely used in video coding standards such as H.263 and MPEG-4 [5]. The conventional floating-point DCT (*FloatDCT*) contains substantial floating-point operations, especially the multiplications which require heavy computation and power resources in mobile devices. An Integer DCT (*IntDCT*) [6] method that can greatly increase the efficiency is adopted in the proposed codec. The implementation of IntDCT is in the form of shift operations and additions, and all internal nodes have finite precision. An 8-pt IntDCT scheme with a complexity of 45 additions and 18 shift operations provides comparable performance to FloatDCT and is suitable for mobile scenarios.

In the proposed codec, the forward IntDCT is used in the encoder followed by the inverse IntDCT in the decoder. The PSNR degradation of this scheme is only about 0.2 to 0.5 dB in contrast to FloatDCT and inverse FloatDCT, whereas about 2.6 to 3.5-fold speed increase is achieved as shown in Table 3.

Table 3: Average PSNR degradation and speed improvement ratio of DCT/IDCT module for the QCIF samples coded at 15 fps and 56 kbps

	PSNR loss (dB)	Speed improvement ratio
Miss_am	0.53	2.86
News	0.42	3.54
Carphone	0.30	2.59
Coastguard	0.23	2.62

2.3 DCT/Quantizer Bypass Algorithm

Besides the motion estimation and DCT components, quantization which involves multiplications also requires significant computations. It is possible to skip some of the DCT and quantization calculations [7] [8]. The DCT/Quantizer

Bypass algorithm is a straightforward and efficient method to reduce computations with virtually no visible loss in video fidelity.

The algorithm is based on two observations: first, for most sequences at low bit rates, a significant portion of MBs have DCT coefficients that are all reduced to zero after quantization; secondly, in the majority of the inter-macroblocks, the DC coefficient of DCT has a larger magnitude than all the other coefficients in a transformed block. Since the coefficients in a block will quantize to zeros only if their magnitudes are less than 2Q, where the variable Q is the quantization parameter, it is possible to predict all-zero quantized (AZQ) MBs using the DC coefficient, and the DCT and quantization calculations associated with those MBs can be eliminated. A substantial saving can be achieved in this way.

In the original algorithm, a macroblock is regarded as AZQ if all DC coefficients of four 8x8 luminance blocks in the MB have magnitudes less than 2Q. As concluded in that paper, under a few conditions, the fidelity degradation is visible and manifests itself as chrominance changes while no distortion appears in the luminance. This is because AZQ blocks are predicted solely on luminance values, which may be an inaccurate indicator for chrominance blocks. In our proposed scheme, chrominance values are taken into account to make the criterion more stringent. Since the DC term corresponds to 1/8 sum of the values in the 8x8 block, two criterions are defined as follows.

Criterion1: $Sum_{LumBlock} < \alpha \times Q$

Criterion2: $Sum_{ChromBlock} < \beta \times Q$

A macroblock is identified as AZQ only when all four luminance blocks satisfy criterion 1 and two chrominance blocks satisfy criterion 2. Suitable parameters in practice are α=8 and β=16, which tighten the condition slightly. Although these more stringent criteria will miss some macroblocks that should be identified as AZQ, they also reduce the number of incorrectly selected macroblocks, which can degrade the final picture fidelity. Meanwhile, the chrominance quality can also be preserved.

The DCT/Quantizer bypass process is only performed on inter-macroblocks, because intra-macroblocks are less likely to satisfy the condition that the DC coefficient has the largest magnitude.

Table 4 shows that substantial DCT and quantization calculations are skipped while the PSNR degradation is marginal. In the table, a negative degradation actually means PSNR improvement. This is possible because not only computations are reduced but also the bits required to code those macroblocks are reduced, and then the saved bits improve the overall PSNR.

Table 4: Percentage of DCT/Quantizer bypassed and average PSNR degradation for the QCIF samples coded at 15 fps and 56 kbps

	Bypassed	PSNR degradation (dB)
Miss_am	17.64%	0.02
News	35.10%	0.14
Carphone	10.55%	-0.03
Coastguard	8.74%	-0.03

3. EXPERIMENTAL RESULTS

To examine the effectiveness of the aforementioned algorithms on reducing computation time and preserving video quality, we chose sequences with vastly varying content from standard MPEG-4 test video clips to test the performance of the overall encoder. The results of the Miss_am and Carphone sequences are shown in this paper. The Miss_am sequence represents scenes with little head-and-shoulder movements and a still background. The Carphone sequence has large facial motion and a fast moving background. Both sequences are in QCIF format and encoded at 15 frames per second and 56 kbps.

Figure 5 shows the PSNR performance of the proposed encoder and an unoptimized encoder which uses full search on motion estimation and the FloatDCT without DCT/Quantizer bypass method. From the experimental result, the two PSNR curves are close in each figure and the proposed encoder provides relatively satisfactory performance in different video sequences with low and high motion activities.

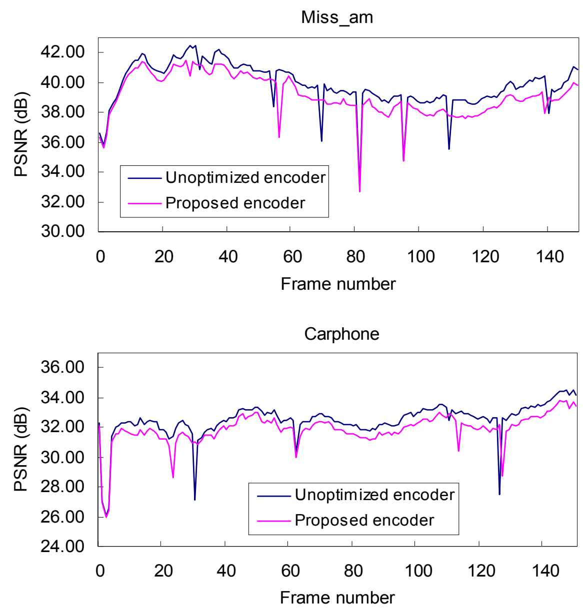

Figure 5: PSNR comparison between the unoptimized encoder and the proposed encoder, average PSNR degradations are 0.68 dB and 0.47 dB respectively.

Figure 6: Miss_am frame number 50 and carphone frame number 58 coded by unoptimized encoder (a) (c) and by proposed encoder (b) (d), PSNR degradations are 0.51 dB and 0.45 dB respectively.

Four reconstructed frames encoded by the unoptimized encoder and by the proposed encoder are shown in Figure 6 for subjective evaluation. No significant subjective degradation is present using the proposed encoder.

The unoptimized encoder and the proposed encoder were also tested on mobile devices. Table 5 shows the maximum frames rates achieved on an iPAQ 3650 Pocket PC which possesses a 206 MHz StrongARM processor and 32MB RAM. From the experimental results, we can see that the computational cost has been significantly reduced using the proposed methods and the resultant frame rate is worthy of consideration for mobile devices.

Table 5: Maximum frame rates (fps) obtained on iPAQ 3650

	Unoptimized encoder	Proposed encoder
Miss_am	1.42	10.93
News	1.45	12.04
Carphone	1.3	10.12
Coastguard	1.11	9.86

4. A PRACTICAL VIDEO COMMUNICATION IMPLEMENTATION ON MOBILE DEVICES

We developed a video communication system, for both PCs and mobile devices using the complexity-reduction algorithms presented in the previous section. Currently, the Mobile Device versions support the majority of the Handheld PCs and Pocket PCs which are based on StrongARM, XScale, MIPS or SH3 processors, and support several types of digital cameras, including HP Jornada pocket camera, FlyCAM-CF camera, FlyJacket iCAM, and Pretec CompactCamera.

The system is composed of three layers: interface layer, service layer and network layer as shown in Figure 7. The aforementioned video compression and decompression techniques are employed in the service layer, and serve as a core component of the system. The ILS (Internet Locator Service) component is used to find other online users, and the Microsoft .NET Messenger Service component and the SIP (Session Initiation Protocol) component are used to manage a buddy list and obtain buddy presence information. The three components can effectively facilitate video communication among users.

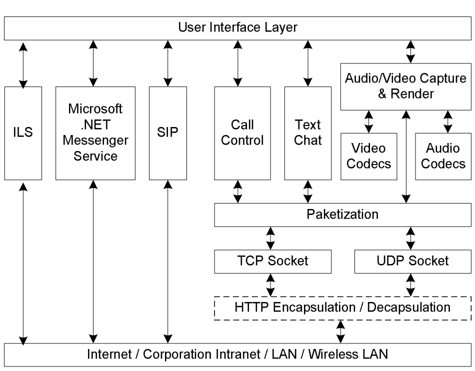

Figure 7: Video communication system architecture

Table 6 shows the frame rates achieved using this system on different mobile devices and different cameras. These frame rates are less than that shown in Table 5, because besides compressing pictures using the proposed encoder, there are several primary time-consuming processes, including capturing video frames, performing color space conversions and transporting compressed bitstream via networks. In practice, the image resolution is 160x120 and the target bit rate is 56kbps. Results show that smooth real-time video communication can be realized by using the proposed system.

Table 6: Maximum frame rates (fps) obtained on different devices

Mobile device	HP camera	FlyCAM-CF	FlyJacket iCAM
iPAQ 3650 Pocket PC	6.6	6.3	7.7
HP Jornada 720 Handheld PC	7.0	Not applicable	Not applicable

5. CONCLUSIONS

In this paper, a low-complexity video coding scheme is proposed and a practical video communication implementation is presented.

Similar to the H.263 encoder, motion estimation, DCT and quantization are the three primary computation-consuming modules in the video encoder. In our architecture, the complexity of motion estimation is significantly reduced by the predictive algorithm in comparison to other fast motion estimation methods, the integer DCT approach is much faster than conventional floating-point DCT, and the DCT/Quantizer bypass algorithm can substantially save resources by skipping some of the DCT and quantization calculations. Experimental results show that the PSNR degradation is small and significant computation reduction is achieved. The proposed techniques have applications where low-complexity computing is required, and is especially suitable for mobile devices.

Future directions may include offering error resilience for wireless or unreliable packet-based transport networks and improvements in compression efficiency and picture quality.

6. ACKNOWLEDGMENT

The authors thank Steve Lin for proofreading the paper.

7. REFERENCES

[1] ITU-T Recommendation H.263 Video coding for low bit rate communication, 02/98.

[2] S. Zhu and K.K Ma, A new diamond search algorithm for fast block-matching motion estimation, IEEE Trans. On Image Processing, vol. 9, no. 2, pp.287-290, Feb 2000.

[3] C.H. Cheung, L.M. Po, A Novel Small-Cross-Diamond Search Algorithm for Fast Video Coding and Videoconferencing Applications, ICIP 2002, pp.681-684.

[4] A. Chimienti, C. Ferraris and D. Pau, A Complexity-Bounded Motion Estimation Algorithm, IEEE Trans. On Image Processing. vol. 11, no. 4, pp.387-392, Apr 2002.

[5] ISO/IEC JTC1/SC29/WG11 N3312 Coding of moving pictures and audio March 2000/Noordwijkerhout.

[6] Y.J. Chen, S. Oraintara and T. Nguyen, VIDEO COMPRESSION USING INTEGER DCT, ICIP 2000, Vancouver, Canada.

[7] A. Yu, R. Lee and M. Flynn, Performance Enhancement of H.263 Encoder Based On Zero Coefficient Prediction, ACM Multimedia 97, Seattle, USA, pages 21-29.

[8] M.T. Sun, I.M. Pao, Statistical Computation of Discrete Cosine Transform in Video Encoders, Journal of Visual Communication and Image Representation, vol. 9, no. 2, Jun 1998, pp.163-170