Adaptive Algorithm To Reduce Computational Complexity In Video Coders

A.Shenbagavalli[†], Dr. L.Ganesan[††]

National Engineering College Kovilpatti, TamilNadu, India.[†]
Prof. and Head/CSE dept Alagappa chettiar college of Engg. and Tech., Karaikudi, India.[††]

Summary
Variable-complexity algorithms provide a means of managing the computational complexity of a software video CODEC. The reduction in computational complexity provided by existing variable-complexity algorithms depends on the video scene characteristics and is difficult to predict. A novel approach to variable complexity encoding is proposed in this paper. A variable complexity DCT algorithm will be updated adaptively in order to maintain a near-constant computational complexity. This adaptive update algorithm will be capable of providing a significant, predictable reduction in computational complexity with only a small loss of video quality. The proposed approach may be particularly useful for software-only video encoding.

Key words:
Motion estimation, motion compensation, motion vector, DCT coefficients, computational complexity.

1. Introduction

DIGITAL video applications are becoming more popular in our everyday lives. Currently, there are several video standards established for different purposes, such as MPEG-1 and MPEG-2 [1] for multimedia applications and H.261 and H.263 [2] for videophone and video-conferencing applications. All these standards use the discrete cosine transform (DCT), motion compensation (which involves motion estimation and motion-compensated prediction), quantization, and variable-length coding (VLC) as building blocks. Using these video-coding standards, video encoders require huge amounts of computation since motion estimation, DCT, and IDCT are all very computationally intensive. Thus, most high-quality video encoders are implemented in hardware that is relatively costly and inflexible. There is significant interest and research in reducing the computations so that a high quality video encoder can be implemented using only software. Previously, the efforts to reduce the computations of video encoders were mainly focused on the fast motion-estimation algorithm. However, as the motion-estimation algorithm becomes optimized, to speed up the video encoders further we also need to optimize other functions, such as DCT and inverse DCT (IDCT).

Manuscript received July 5, 2008.
Manuscript revised July 20, 2008.

2. Theoretical Background

This section provides a very brief overview of video compression and video compression standards. The limited space precludes a detailed discussion; however we highlight some of the important principles and practices of current and emerging video compression algorithms and standards that are especially relevant for video communication and video streaming. An important motivation for this discussion is that both the standards MPEG-1/2/4 [1] and H.261/3/4 [2] are based on the same basic principles and practices, and therefore by understanding them one can gain a basic understanding for both standards. Another goal of this section is to describe what are the different video compression standards, what do these standards actually specify, and which standards are most relevant for video compression/streaming.

2.1 Introduction

Video has been an important media for communications and entertainment for many decades. Initially video was captured and transmitted in analog form. The advent of digital integrated circuits and computers led to the digitization of video, and digital video enabled a revolution in the compression and communication of video. Video compression became an important area of research in the late 1980's and 1990's and enabled a variety of applications including video storage on DVD's and Video-CD's, video broadcast over digital cable, satellite and terrestrial (over-the-air) digital television (DTV), and video conferencing and videophone over circuit switched networks.

2.2 Brief Overview of Video Compression

Video compression is achieved by exploiting the similarities or redundancies that exists in a typical video signal. For example, consecutive frames in a video sequence exhibit temporal redundancy since they typically contain the same objects, perhaps undergoing some movement between frames. Within a single frame there is spatial redundancy as the amplitudes of nearby pixels are often correlated. Similarly, the Red, Green, and Blue color components of a given pixel are often correlated. Another

goal of video compression is to reduce the irrelevancy in the video signal that is to only code video features that are perceptually important and not to waste valuable bits on information that is not perceptually important or irrelevant. Identifying and reducing the redundancy in a video signal is relatively straightforward, however identifying what is perceptually relevant and what is not is very difficult and therefore irrelevancy is difficult to exploit.

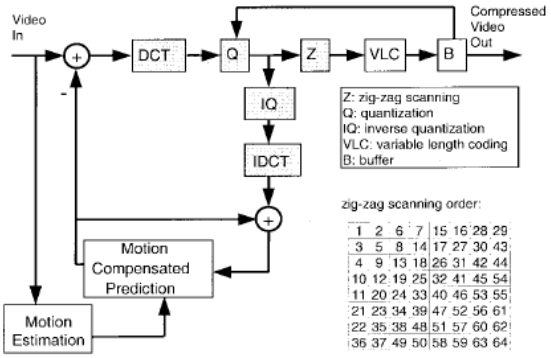

Fig. 1 Block Diagram of Video Encoder

To begin, we consider image compression, such as the JPEG standard, which is designed to exploit the spatial and color redundancy that exists in a single still image. Neighboring pixels in an image are often highly similar, and natural images often have most of their energies concentrated in the low frequencies. JPEG exploits these features by partitioning an image into 8x8 pixel blocks and computing the 2-D Discrete Cosine Transform (DCT) for each block. The motivation for splitting an image into small blocks is that the pixels within a small block are generally more similar to each other than the pixels within a larger block. The DCT compacts most of the signal energy in the block into only a small fraction of the DCT coefficients, where this small fraction of the coefficients are sufficient to reconstruct an accurate version of the image. Each 8x8 block of DCT coefficients is then quantized and processed using a number of techniques known as zigzag scanning, run length coding, and Huffman coding to produce a compressed bit stream. In the case of a color image, a color space conversion is first applied to convert the RGB image into a luminance/chrominance color space where the different human visual perception for the luminance (intensity) and chrominance characteristics of the image can be better exploited. A video sequence consists of a sequence of video frames or images. Each frame may be coded as a separate image, for example by independently applying JPEG-like coding to each frame. However, since neighboring video frames are typically very similar much higher compression can be achieved by exploiting the similarity between frames. Currently, the most effective approach to exploit the similarity between frames is by coding a given frame by (1) first predicting it based on a previously coded frame, and then (2) coding the error in this prediction. Consecutive video frames typically contain the same imagery, however possibly at different spatial locations because of motion. Therefore, to improve the predictability it is important to estimate the motion between the frames and then to form an appropriate prediction that compensates for the motion. The process of estimating the motion between frames is known as motion estimation (ME), and the process of forming a prediction while compensating for the relative motion between two frames is referred to as motion-compensated prediction (MC-P) as shown in Fig 1. Block-based ME and MC-prediction is currently the most popular form of ME and MC-prediction: the current frame to be coded is partitioned into 16x16-pixel blocks, and for each block a prediction is formed by finding the best-matching block in the previously coded reference frame. The relative motion for the best-matching block is referred to as the motion vector.

There are three basic common types of coded frames as shown in figure 2.They are (1) Intra-coded frames, or I-frames, where the frames are coded independently of all other frames, (2) Predictively coded, or P-frames, where the frame is coded based on a previously coded frame, and (3) Bi-directionally predicted frames, or B frames, where the frame is coded using both previous and future coded frames.

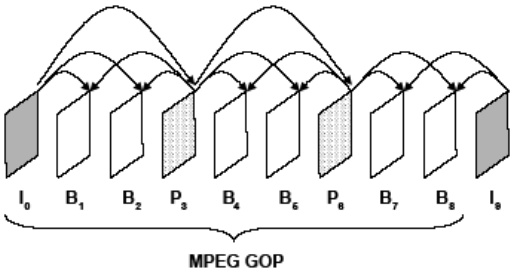

Fig. 2 Example of Prediction Dependencies between frames

Figure 2 illustrates the different coded frames and prediction dependencies for an example MPEG Group of Pictures (GOP). The selection of prediction dependencies between frames can have a significant effect on video compression performance, e.g. in terms of compression efficiency and error resilience. Current video compression standards achieve compression by applying the same basic principles. The temporal redundancy is exploited by applying MC-prediction, the spatial redundancy is exploited by applying the DCT, and the color space redundancy is exploited by a color space conversion. The resulting DCT coefficients are quantized and the nonzero

quantized DCT coefficients are run length and Huffman coded to produce the compressed bit stream.

3. Previous work Result and Discussions

About 10 frames extracted from a video clipping is given as input for the four search algorithms for motion estimation: Three step search, Four step search, Two Dimensional Logarithmic search and binary search [3]. These four algorithms are compared based on their CPU time used for computational complexity. Matlab 6.5 is used for simulation.

Table 1: Computation Time for Search Algorithm

Image Format	No. of Frames	Algorithm	CPU Time (Seconds)
BMP	10	TSS	1.73
		FSS	1.20
		TDLS	0.60
		BS	0.80

4. Computational Complexity

The simplest method of reducing computational complexity is to reduce the frame rate of the video sequence, i.e. to "skip" frames of video. This method tends to produce "jerky" video with a variable frame rate at the decoder. Recently there has been much interest in Variable Complexity Algorithms (VCAs) for video coding. A VCA enables the CODEC to trade-off computational complexity and video quality in a more flexible way than simply by skipping frames. VCAs have been proposed for computationally intensive functions including DCT [4, 5], IDCT and motion estimation [6]. Applying a combination of reduced-complexity algorithms can reduce the computation requirements of MPEG-2 video encoding by over 70% [2]. In general, VCAs enable a reduction in computational complexity at the expense of a loss of image quality. In this paper, we take the following approach to support complexity management for real-time video encoding. The discrete cosine transform (DCT) [7] is identified as a computationally intensive function. Methods of predicting or modeling the output of the DCT (and therefore bypassing some computational steps) are compared. An adaptive algorithm is described and is shown to be suitable for dynamically controlling the complexity of the DCT (and related functions) to maintain a ‘‘target’’ level of computational complexity.

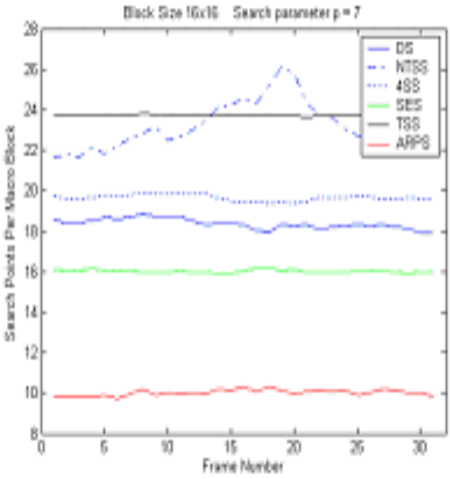

Fig. 3 Search points per macro block while computing the PSNR performance of Fast Block Matching Algorithms

5. DCT Computational Complexity

5.1 Modeling DCT coefficients

The DCT and other related functions like IDCT, quantization and rescaling can consume nearly 1/3 of the processing resources and so it is useful to examine methods of reducing the computational complexity of these functions. The residual energy in many blocks of a typical difference macro block is low, such that most or all of the DCT coefficients for the block are zero after quantization. The computational complexity of the DCT and quantization operations may be significantly reduced by skipping the DCT and quantize operations for blocks that are likely to contain all zero coefficients after quantization. It is possible to predict quantized blocks containing all zero coefficients (i.e. blocks with zero End-Of-Block or EOB). There is a correlation between the Mean Absolute Error (MAE) or Sum of Absolute Differences (SAD) for the residual block and the probability of zero EOB [4]. A low MAE or SAD indicates a high probability of zero EOB. Pao and Sun [4] proposed predicting EOB for each block based on SAD for the current macro block. However, there is a higher correlation between EOB and block SAD than between EOB and macro block SAD.

5.2 Proposed Algorithm

1. Calculate SAD for each block:

$$SAD = (1/b^2) \sum_{i=1}^{b} \sum_{j=1}^{b} |B_{ij} - C_{ij}|$$

where B_{ij} is the pixel in B block in current frame and C_{ij} its counterpart C_{ij} in C of Reference frame motion compensated prediction from the reference frame. (SAD is usually calculated during motion estimation and so this step does not involve any extra computation.)
2. Compare SAD/Quant with T (where Quant is the quantizer step size for the current macro block).
3. If SAD/Quant < T: model predicts that EOB = 0 (do not carry out DCT and quantization, set all block coefficients to zero).
4. If SAD/Quant ≥ T: model predicts that EOB>0 (carry out DCT and quantization for the current block).

5.3 Adaptive control algorithm for DCT function

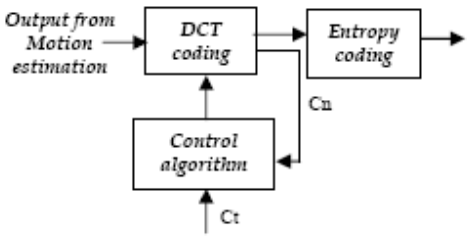

Fig. 4 Complexity Control of DCT

After motion compensation, the residual energy in many blocks is very low. There is a high possibility that most or all of the DCT coefficients of a block become zero after quantization. The computation of the DCT and quantization functions may be significantly saved by skipping the DCT and quantization operations for blocks that are likely to contain all zero coefficients after quantization. We develop a feed back control algorithm shown in Figure 3 to decrease operation time of DCT function. Ct is target computational complexity and Cn is measured complexity of frame n. The aim of this algorithm is to reduce the complexity to the target level (Ct) and keep it at the level through the entire video sequence. After coding frame n, the computational complexity of frame n (Cn) is feed back to control algorithm. Since the level of detail and motion in a video sequence tends to vary gradually, it is possible for controller to adapt for next frame to achieve target complexity based on the actual complexity of current frame. A detailed mathematical description and of this algorithm can be found in [4] [6].

5.4 Adaptive control algorithm for motion estimation

This adaptive algorithm is based on the Nearest Neighbor Search (NNS) motion estimation algorithm [8]. NNS always searches four neighboring locations in a diamond-shaped layer and the neighbor location that has the minimum SAD is chosen to be the centre point of the next search "layer".

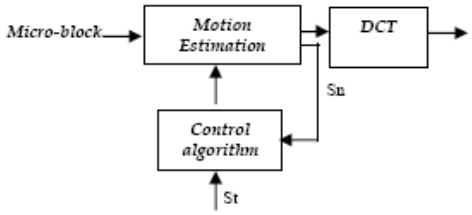

Fig. 5 Complexity Control of Motion Estimation

We propose an algorithm in Figure 3 [9] to update the search layer for the next frame according to the estimated motion estimation complexity (Sn) and search layer of current frame in order to obtain target complexity (St). Experimental results in [9] indicate that the algorithm achieve target complexity throughout the video sequence while rate distortion performance is reduced by around 0.5-0.7dB for a high-movement video sequence and is virtually unchanged for a moderate-movement sequence.

References

[1] J McVeigh et al, "A Software-Based Real-Time MPEG-2 Video Encoder", IEEE Trans. Circuits and Systems for Video Technology 10(7), October 2000
[2] ITU-T Recommendation H.263, "Video coding for low bit rate communication", 1998
[3] Deepak Turaga, Mohamed Alkanhal "Search Algorithms for Block-Matching in Motion Estimation" Mid-Term project 18-899 spring, 1998
[4] I-M Pao and M-T Sun, "Modeling DCT Coefficients for Fast Video Encoding", IEEE Trans. Circuits and Systems for Video Technology, June 1999
[5] K Lengwehasatit and A Ortega, "DCT Computation based on Variable Complexity Fast Approximations", Proc. ICIP98, October 1998
[6] M Gallant, G Côté and F Kossentini, "An Efficient Computation-Constrained Block-Based Motion Estimation Algorithm for Low Bit Rate Video Coding", IEEE Trans. Image Processing 8(12), December 1999
[7] Zhongde Wang "Pruning the Fast Discrete Cosine Transform" IEEE transactions on communications, vol. 39, NO. 5, MAY 1991
[8] Iain E G Richardson and Yafan Zhao, "Adaptive management of Video Encoder complexity", Journal of Real-time Imaging (accepted in 2001)
[9] Y Zhao and I E G Richardson, "Computational Complexity Management of motion Estimation in Video Encoders", Proc. DCC02, April 2002.