MPEG−2 to H.263 Transcoder System Modeling and Implementation

Kevin Baldor
Sue Baldor

May 8, 2002

Abstract

In this paper, we discuss the implementation and modeling of an MPEG−2 to H.263 transcoder. With the emergence of video on demand in the cable television market, broadcasters are considering expanding this service to the internet. Due to the limited bandwidth of internet communication, the high quality and high bandwidth broadcasts must be reduced to a low bitrate suitable for video transmission over the internet. For this purpose, we focus on transcoding between the MPEG−2 and H.263 video standards. Our implementation focuses on decreasing the spatial resolution of a frame in the video sequence. Our model of the transcoder introduces parallelism to help facilitate real−time goals.

Introduction

High definition television (HDTV) and digital cable provide high quality digital video to consumers at home. However, the bitrates required can be fairly high. Both markets utilize the MPEG−2 standard, which supports bitrates as low as 1.5 Mbps. Current HDTV formats require between 16.9 and 18.8 Mbps depending on the resolution used, while current digital cable boxes support up to 30Mbps [2]. Video transmission over the internet will require much lower bitrates, with even broadband connections supporting typical bitrates of only 200 to 300 kbps.

Transcoding is the discipline concerned with conversion from one standard to another efficiently. Two video coding standards, MPEG−2 and H.263 contain many similarities than can be leveraged to reduce the complexity of the conversion. Most significantly, they both use similar frame types. Each standard supports I−, P−, and B− frames. I−frames (intra−frames) apply the Discrete Cosine Transform (DCT) to non− overlapping blocks of pixels (macroblocks) and quantize the resulting coefficients. I− frames are self−contained; in other words, they do not depend upon any other frame to define their appearance. P−frames use blocks from the previous frame to define its blocks, while B−frames use the blocks from the preceding and following I− or P−frame. For enhanced compression, H.263 has a special PB frame type that combines a P− and B−frame [3].

In this paper, we incorporate parallel processing techniques with common methods utilized in transcoding implementations.

System Models

The high level SDF graph demonstrates the parallelism of the MPEG standard. Each X−Code block operates upon a set of frames which lie between two intra−coded frames in the input stream. The purpose of this model is to analyze the tradeoffs associated with a parallel processor implementation of our transcoder.

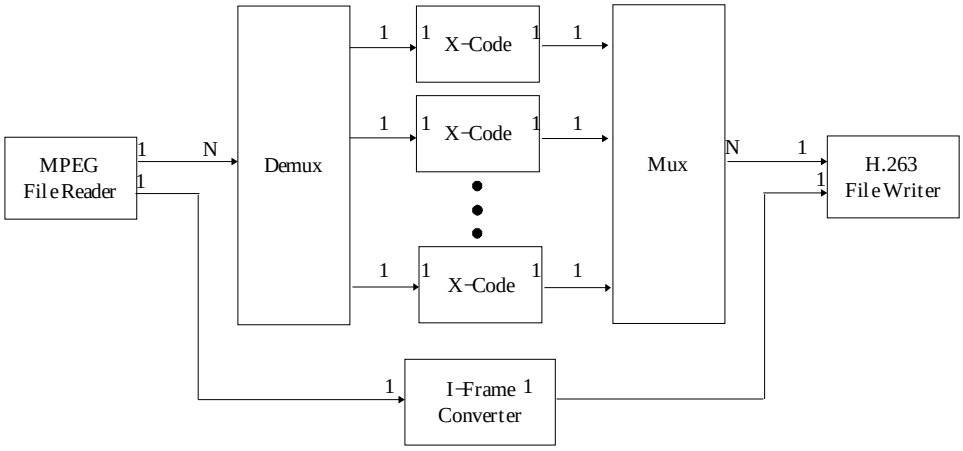

Figure 1. SDF Model of System

The model for the individual X−code block reuses the block which converts a group of frames containing two B−frames and one P−frame to a PB−frame to reduce code size. The details inner workings of each block will be discussed shortly.

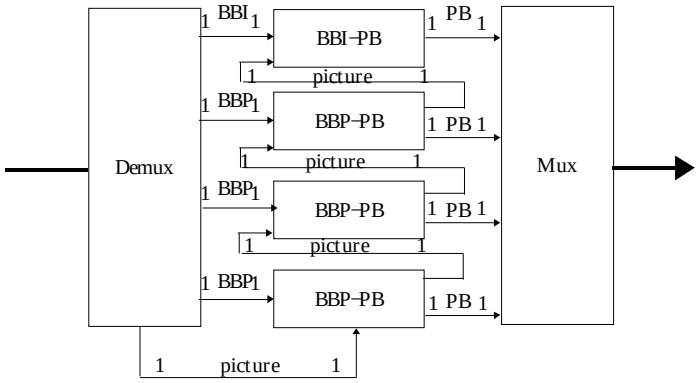

Figure 2. SDF Model of Transcoder

Resolution Reduction

The primary way we reduce the bitrate of the video stream is by reducing the resolution of each frame. Resolution reduction involves combining several blocks to produce a single block in the output stream. In our case, we are reducing the resolution in the x direction by a factor of two and the resolution in the y direction by a factor of two. Thus, we combine four macroblocks from an MPEG−2 frame to produce a single macroblock in an H.263 frame. This operation requires two primary operations: recalculating DCT coefficients and recalculating motion vectors.

To obtain the new DCT coefficients, we use a method which performs decimation in the DCT domain. In this method, each of the four 8x8 DCT blocks is converted into a 4x4 block by removing all but the upper left 4x4 terms (those corresponding to low spatial frequencies). This exploits the fact that most natural visual information is contained in the lower spatial frequencies. These 4x4 DCT blocks are then inverse DCT transformed and combined to produce an 8x8 block, which is DCT transformed for transmission [5]. Figure 3 depicts how the blocks are combined.

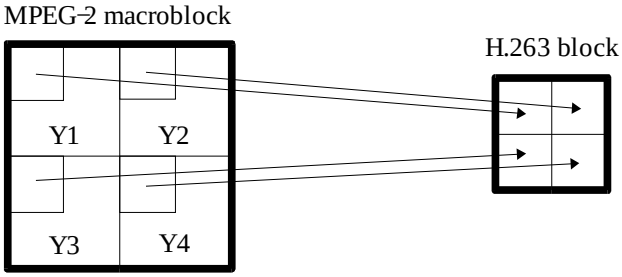

Figure 3. Combining DCT Blocks

The other important operation needed for resolution reduction is the recalculation of the motion vector. The simplest approach would be to decompress the video stream and perform a full motion estimation operation. This approach is extremely computationally intensive due to the massive number of difference metric calculations required. Instead, we adapted a method proposed in [5] which uses the existing motion vectors from the MPEG−2 video stream to form an estimate of the motion vectors for the downscaled video stream.

In our MPEG−2 video stream, each macroblock contains a single motion vector. Similarly, our H.263 video stream contains a single motion vector for each macroblock. Therefore, we need a method to take the four MPEG−2 motion vectors and produce a single H.263 motion vector, as depicted in Figure 4.

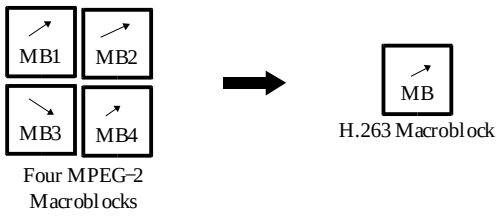

Figure 4. Illustration of Motion Vector Calculation

One approach would be to simply average the four motion vectors to produce the new one. This is correct when the motion vectors are similarly aligned, but incorrect if the the four motion vectors are poorly aligned. Therefore, a spatial activity measurement is used to make an estimate of the motion vector.

The spatial activity measurement is calculated by summing up the number of non−zero AC coefficients of the residual DCT coefficients in each block of the macroblock. The resulting metric has a higher value for macroblocks with larger prediction error because they contain greater energy in their residual DCT blocks. This

will weight the estimate toward the worst prediction. The estimated motion vector, $\overline{mv}$, is derived from the four original motion vectors as follows:

$$\overline{mv} = \frac{1}{2} \frac{\sum_{i=1}^{4} \overline{mv}_i A_i}{\sum_{i=1}^{4} A_i}$$

Each of the $\overline{mv}_i$, is one of the original motion vectors, and A_i is the activity measurement of macroblock *i*. If the sum of the four activity measurements becomes zero, we revert to a simple average.

Frame Type Conversion

We have set our MPEG−2 stream to consist of groups of frames of the following frame types in order: IBBPBBPBBPBBI. The output H.263 stream consists of a single I−frame followed by an unlimited number of PB−frames. These differences in the streams create the need for two major frame type conversions [4].

We chose to convert a BBP group of frames from the MPEG−2 stream into a single PB−frame in the H.263 stream. This conversion prevents the conversion of a B−frame into a P−frame. Such a conversion is a potential source of error because future frames are produced using its contents [1]. Whereas our method introduces error in intermediate frames, these errors will not propagate .

The BBP (MPEG−2) to BP−frame (H.263) conversion is a two step process. First the two B−frames from the MPEG−2 stream are merged together to produce one B−frame. The motion vectors are calculated using a similar technique to the one used in resolution reduction. However, due to the temporal shift in the resulting B−frame, the components of the forward and backwards vectors must be scaled accordingly. The following set of equations define how the new forward and backward motion vectors are

calculated:

$$\overline{mv_f} = 0.75\, A_1\, \overline{mv_{f1}} + 1.5\, A_2\, \overline{mv_{f2}}$$

$$\overline{mv_b} = 1.5\, A_1\, \overline{mv_{b1}} + 0.75\, A_2\, \overline{mv_{b2}}$$

Each A_i the energy calculation, as before. Each $\overline{mv_b}$ denotes a backwards motion vector, and each $\overline{mv_f}$ denotes a forwards motion vector. The first term in the sum uses the motion vector from the first B−frame, and the second term uses the motion vector from the second B−frame.

After merging the two MPEG−2 B−frames, the transcoder reduces the resolution in the resulting B−frame and following P−frame. Using the same method as before, the motion vectors for each new macroblock are estimated. Additionally, the DCT coefficients for each macroblock in the P−frame are calculated. However, the DCT coefficients from the macroblocks in the B−frame are thrown away. The forward motion vectors from the reduced P−frame are kept, and the backward motion vectors from the reduced B−frame are used to calculate the delta motion vectors for the H.263 PB−frame.

The second frame type conversion necessary involves converting the MPEG−2 I−frame into an H.263 P−frame. During this process, we must ensure that we conform to the H.263 standard's rules for I−block transmission. According to the standard, each block must be updated by sending an intra−coded block (I−block) every 132 frames. We conform to this requirement through an updating policy rather than keeping track of which blocks have been updated. In our updating policy, every 15 macroblocks are forced to be updated (intra−coded) during the I−frame conversion process. The macroblocks are assigned for updating beginning at some offset which is incremented for each group of frames. Each updated macroblock in the H.263 P−frame corresponds to

four intra−coded macroblocks from the MPEG−2 I−frame. Spatial resolution is scaled down using methods previously discussed.

Motion vectors and residual DCT blocks must be calculated for macroblocks which are not updated. We do not search the full search area for the motion estimates. Instead we use a starting "guess" and limit ourselves to searching out to two pixels in each direction (a 5x5 search area). For our guesses, we use the forward motion vectors from the previous downscaled B−frame. The sum of absolute differences (see equation below) is computed at each point within the search area.

$$SAD = \sum_{i=0}^{N} \sum_{j=0}^{N} \left| p_{ij} - p'_{ij} \right|$$

The point at which the SAD is minimized becomes our match. The resulting motion vector is the difference between the current position and the position to the minimum SAD point. The residual DCT coefficients are calculated using the computed motion vector and decoded and downscaled intra−coded macroblocks.

Conclusion

From this project, we have learned a great deal about differences and similarities between the MPEG and H.263 standards and the issues involved in transcoding between them. From our work, it appears possible to implement a transcoder which operates in real−time through a parallel system as long as a delay is tolerable. Also, we have found that the workload required for video transcoding is not significantly higher than that of a decoder. This property results primarily from the reuse of the motion vectors from the input stream. Since the search for these vectors is computationally costly, this produces a significant performance boost.

References

[1] N. Bjork and C. Christopoulos, "Transcoder Architectures for Video Coding," *IEEE Transactions on Consumer Electronics,* vol. 44, no. 1, Feb. 1998.

[2] D. Cervenka. "HDTV Vortex of Uncertainty for Cable", http://www.cedmagazine.com/ced/9808/9808f.htm, Communications Engineering and Design, 1998.

[3] K. Jack, "Video Demystified: A Handbook for the Digital Engineer", High Text Publications, Solana Beach, CA 2001.

[4] T. Shanableh and M. Ghanbari, "Heterogeneous Video Transcoding to Lower Spatio−Temporal Resolutions and Different Encoding Formats, "*IEEE Transactions on Multimedia,* vol. 2, no. 2, pp. 101−109, June 2000.

[5] B. Shen, I.K. Sethi, and B. Vasudev, "Adaptive motion−vector resampling for compressed video downscaling," *IEEE Transactions on Circuits and Systems for Video Technology,* vol. 9, no. 6, pp.929−936, Sept. 1999.