Transcoder Architectures and Techniques

Kevin Baldor
Sue Baldor

Abstract:

In this paper we review the current state of heterogeneous transcoding techniques as applied to an MPEG-2 to H.263 transcoder. Among the techniques considered are format conversion considerations related to differing frame types and the requisite conversions, and bitrate reduction techniques. Since we are focusing on video over handheld wireless devices, we focus upon methods used to reduce the spatial resolution of the video sequence. We also discuss our proposed implementation of an MPEG-2 to H.263 transcoder's model of computation as well as some discussion of parallelization to meet real-time goals.

Introduction:

High definition television (HDTV) will provide high quality digital video to the realm of broadcast television. However, the bitrates required will be fairly high. HDTV will be based on the MPEG-2 standard, which supports bitrates as low as 1.5 Mbps, but the "High Level" version intended for HDTV has a maximum bitrate of 80 – 100 Mbps [4]. Currently planned formats require between 16.9 and 18.8 Mbps depending on the resolution used [2].

For video over handheld devices, this bitrate must be reduced to the range supported by third generation wireless, 2 Mbps down to 128 Kbps for vehicles. To accomplish this, we propose that the MPEG datastream should be converted to the low-bandwidth capable standard H.263, which was initially restricted to bitrates of 64 Kbps or lower.

Transcoding is the discipline concerned with conversion from one standard to another efficiently. The two standards, MPEG-2 and H.263, contain many similarities that 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 are those which use Discrete Cosine Transform based compressed blocks to represent the frame. I-frames are self-contained, i.e. 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 into one unit [4].

In this literature review, we present some of the previous work on transcoding and related technologies relevant to our goal of producing a specialized real-time MPEG-2 to H.263 transcoder for wireless transmission of high definition broadcast television.

Rate Reduction:

Rate reduction is achieved by increasing the quantization of the encoder in the transcoder. This action is coupled with passing the motion vectors and macroblock information directly from the decoder to the encoder, rather than using that information to completely decode the bit stream. However, the passed motion vectors must be refined and the passed macroblock information must be re-coded [1].

On average, the computing complexity improved 39% (over a basic video transcoder) with the rate reduction methods [1]. However, the rate reductions come at a significant loss of quality. Furthermore, for our purposes, the screens will be so small that resolution reduction will be required for display, so it makes sense to take advantage of the bitrate reductions which would result from reducing the resolution prior to transmission.

Resolution Reduction:

Resolution reduction involves combining several blocks to produce a single block in the output stream. Accordingly, the computational complexity as compared to the straightforward transcoding implementation at 23% is not as great as rate reduction, but the reduction in quality is lower [1]. When discussing methods here it is assumed that that the resolution is being reduced by a factor of two along each axis.

Several methods are considered for resolution reduction on I-frames. The conceptually simplest involves performing the Inverse Discrete Cosine Transform (IDCT) on four 8x8 blocks to produce a single 16x16 block. This block is then filtered, downsampled, and DCT transformed to produce an 8x8 block.

A less simple conceptually, but somewhat simpler computationally, method 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 then combined to produce an 8x8 block, which is DCT transformed for transmission [5].

Once the resolution is reduced, the motion vectors must be adjusted, but still their reuse decreases computational complexity by more than a factor of three without significantly degrading the quality of the calculation. The method proposed in [6] does not decompress the video stream, but uses existing motion vectors to calculate the new motion vectors for the downscaled video stream. Thus, the time intensive motion estimation operation is avoided.

An $N \times N$ macroblock in the original video stream corresponds to a $N/2 \times N/2$ macroblock in the output video stream. 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 (as in the left side of Figure 1), but incorrect if the four motion vectors are not similarly aligned (as in the right side of Figure 1). An adaptive approach to the problem is a better solution. The authors propose using spatial activity measurement is used to make an estimate of the actual motion vector.

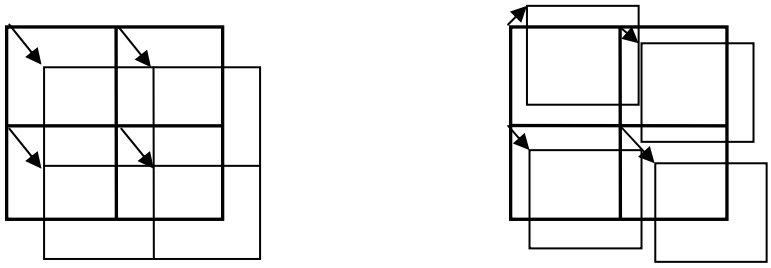

Figure 1: Similarly and Dissimilarly Aligned Motion Vectors

The estimated motion vector is weighted toward the motion vector with the larger prediction error, or toward the worst prediction. The estimated motion vector, $\bar{v}'$, is derived from the four original motion vectors as follows:

$$\bar{v} = \frac{1}{2} \frac{\sum_{i=1}^{4} \bar{v}_i A_i}{\sum_{i=1}^{4} A_i}$$

Each of the $\bar{v}_i$ is one of the original motion vectors, and A_i is the activity measurement of block i in the original video sequence. Each A_i is calculated from the DCT coefficients of block i. In particular, the authors add the number of nonzero AC coefficients. Another proposed method is to sum the absolute value of the AC coefficients.

The method was simulated on three different video sequences, transcoding from a higher bit rate MPEG 1 video stream to a lower one. The peak SNR difference (with respect to the straightforward approach) averaged around 1 decibel. The complexity decreased from 75 million operations per P-Frame to around 20 million operations [6].

T. Shanableh and M. Ghanbari also presented is the improvement in PSNR associated with motion vector refinement ranges from ±.5 pixels through ±15.5. Notably, the improvement beyond ±. 5 pixel, refinement is negligible which suggests a significant reduction in complexity through the reuse of motion vectors even when refinement is applied [5].

Frame Type Conversion:

While the two standards share many similarities in their frame types, they exhibit several significant differences. Chiefly, MPEG places no restrictions upon the arrangement of I-,P-, and

B-frames while H.263 is more restrictive. The result is a need to convert from each frame type to the others.

To simplify the discussion of encoding format conversion, the MPEG stream is assumed to consist of groups of frames of the following types in the order: IBBPBBPBBPBBI. The output H.263 stream formats considered were: a single I-frame followed by an unlimited number of P-frames, and a single I-frames followed by an unlimited number of PB-frames [5].

Since we intend to implement the PB-frame method, if we were to convert all frames we would need to consider the conversion of P-frames to B-frames and vice-versa. Since motion vectors in various directions define these frame types, the motion vector in the output frame must be produced by some combination of the motion vectors of the input file. In the case of P-frames being produced from B-frames, the backward motion vector must be produced from some combination or reversal of the motion vectors of the corresponding blocks from the B-frame in question, the preceding or following B-frames and P-frames. Each of these must be compared by computing a measure of the quality of the match.

However, it would seem that the conversion of a B-frame into a P-frame will probably introduce error, because future frames are produced using its contents. Accordingly, we will avoid this potential source of error by modifying the frame rate such that no P-frames are produced from B-frames. This will increase the complexity of the production of the bi-directional component of the PB-frames, and likely introduce error in these intermediate frames, but those errors will not propagate. The conversion of the input stream to an output stream is illustrated in figure 2. on the next page.

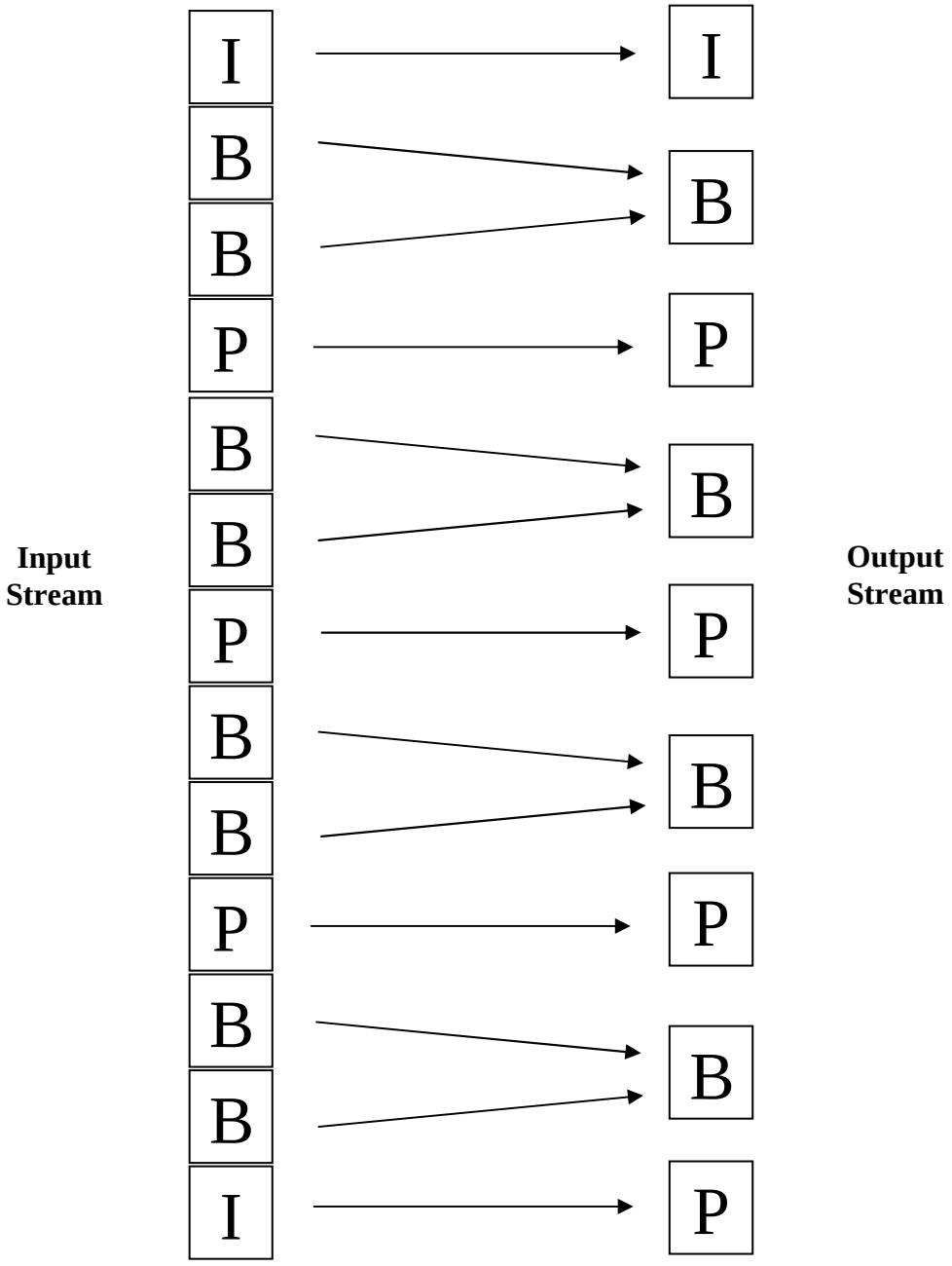

Figure 2: Input to Output Frame Type Conversions

Our Implementation:

The model of computation which we will apply this problem is Synchronous Dataflow. We will assume the same frame orientation as that used in the papers considered, i.e. an input MPEG-2 stream of the form IBBPBBPBBPBBI… and an output H.263 stream of the form

IBPBPBP… . For this system, the SDF graph that describes its continuous operation is shown in Figure 3:

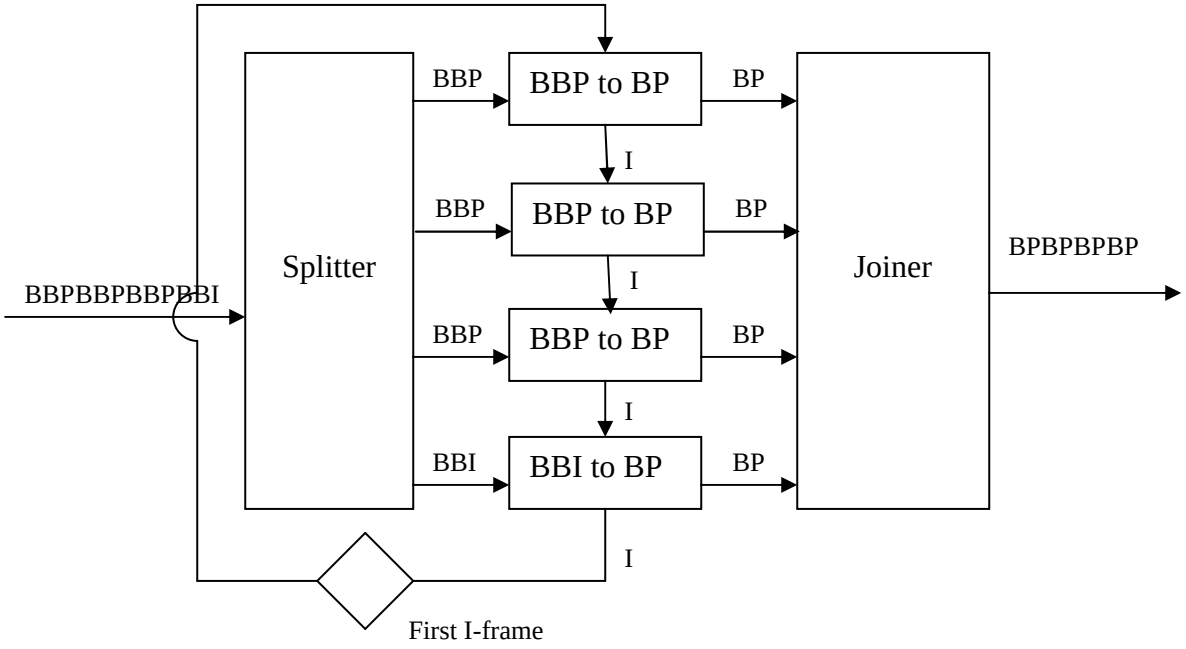

Figure 3: SDF model of computation

Conclusion:

In our project, we will apply the principles of motion vector reuse and spatial resolution reduction presented in the reviewed articles. We anticipate that the SDF graph above will not operate in real time on a single processor. Therefore to accomplish real-time transcoding, we intend to exploit the fact that one set of frames between two I-frames is independent of any other set of frames between a different set of I-frames to produce a parallel solution. This will operate somewhat like the instruction pipelining used on DSP processors to mask the delay associated with commands that require more than one clock cycle to complete.

References

[1] N. Bjork and C. Christopoulos, “Transcoder Architectures for Video Coding,” *IEEE Transactions on Consumer Electronics*, Vol. 44, No. 1, February 1998.

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

[3] P. Cherriman, "ITU H.263 Video Coding", http://www-mobile.ecs.soton.ac.uk/peter/h263/h263.html, 1999

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

[5] 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.

[6] 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, September 1999.

[7] J. Wagner, "Getting to Know Your 3G", http://www.internetnews.com/wireless/article/0,,10692_964581,00.html , Wireless News, 2002