An MPEG-2 to H.263 Transcoder

Nick Feamster[1,2] and Susie Wee[2]

[1]Massachusetts Institute of Technology, Cambridge, MA
[2]Hewlett-Packard Laboratories, Palo Alto, CA

ABSTRACT

We present an MPEG-2 to H.263 transcoder that accepts an interlaced MPEG-2 bitstream as the input and produces a lower-bitrate progressive H.263 bitstream as the output. As both DVD and digital television may use MPEG-2 interlaced sequences, a potential application of such a transcoder is the transmission of a digital television signal over a wireless medium. Another application is transcoding interlaced DVD content for use on lower-resolution thin clients with progressive displays. The proposed algorithm exploits the properties of the MPEG-2 and H.263 compression standards to perform interlaced to progressive (field to frame) conversion with spatial downsampling and frame-rate reduction in a CPU and memory efficient manner, while additionally minimizing picture quality degradation as measured by PSNR. This is the first algorithm to our knowledge that effectively uses both spatial and temporal downsampling in an MPEG-2 to H.263 field to frame transcoder in order to achieve substantial bitrate reduction. This paper discusses recoding experiments used to determine appropriate source and target coding parameters for the transcoder, provides a detailed description of the transcoding algorithm, and describes the performance of a software implementation of the transcoder.

1. INTRODUCTION

Video communication requires the seamless delivery of video content to a broad range of users with different bandwidth and resource constraints. However, the video attributes and compression standards used by the source signal, the communication channel, and the client device are far from seamless. Thus, efficient transcoding algorithms must be designed to mend these mismatches and provide users with a seamless experience. A likely scenario involves the MPEG-2 and H.263 standards. MPEG-2 was designed for high-quality, high-rate applications and is used in digital television and DVD. H.263 was designed for low-rate communication over ISDN and analog telephone lines. An MPEG-2 to H.263 transcoder enables the transmission of MPEG program material over lower-rate communication channels such as ISDN lines, analog telephone lines, wireless links, and other low bandwidth network channels.

In this paper, we present a transcoding algorithm that converts an MPEG-2 input bitstream to a lower-bitrate H.263 bitstream in a manner which both preserves video quality and efficiently utilizes CPU cycles. Bitrate reduction is achieved with spatial and temporal downsampling and requantization, and the details of the MPEG-2 and H.263 compression standards are exploited when performing interlaced to progressive (or field to frame) conversion. This algorithm can be used to transcode bitstreams with a wide variety of input and output bitrates and spatial and temporal resolutions. To simplify the presentation, we discuss two particular transcoders which demonstrate interesting aspects of the proposed algorithm. The first transcoder accepts as input a progressive 30 fps CIF (352×240) MPEG-2 bitstream and produces as output a progressive 10 fps CIF H.263 bitstream, with input and output bitrates of 1.5 Mbps and 500 kbps. The second transcoder accepts an interlaced 30 fps CCIR-601 (720×480) MPEG bitstream as input and produces a progressive 10 fps CIF H.263 bitstream, with input and output bitrates of 5 Mbps and 500 kbps. The general algorithm, however, can be used to process bitstreams with other input and output parameters.

Previous transcoding work has examined tradeoffs in picture quality and computational complexity when transcoding for bitrate and/or resolution reduction within the MPEG standard or within the H.261/3 standard.[1–3] To the authors knowledge, the only work reported on transcoding between the two standards was in[4] where an MPEG to H.263 transcoder was developed for progressive input and output bitstreams when retaining the full temporal frame rate of the video. Our work differs in that our algorithm supports field-coded interlaced video bitstreams and it achieves bitrate reduction by allowing temporal frame rate reductions in addition to spatial resolution reduction and requantization. A high-level description of our algorithm was given in[5]; in this paper we discuss recoding issues that arise when determining the source and target parameters for the transcoder and we provide a more in-depth description of the algorithm and our experimental results.

This paper begins by describing the recoding experiments which were performed to quantify the quality losses which may result from the recoding cycle. Section 3 outlines differences between the MPEG and H.263 standards. Section 4 describes the algorithm itself, including its evolution, as well as its methods of dealing with differences between standards. Section 5 summarizes experimental results and compares the results of our algorithm to a conventional transcoding approach, and section 6 presents conclusions and suggestions for further work.

2. DESIGN ISSUES: RECODING PERFORMANCE AND SOURCE AND TARGET CODING PARAMETERS

In this section, we discuss design issues that arise when developing an MPEG-2 to H.263 transcoder. We then describe the results of our recoding experiments which were used to guide the design process. The first subsection examines the case in which the input and output bitrate remains constant, the second second subsection examines the case in which the output bitstream has a lower bitrate, spatial resolution, and temporal frame rate.

The design of an MPEG-2 to H.263 transcoding algorithm involves determining appropriate coding parameters, such as bitrate and resolution, for the source and target bitstreams. We are interested in designing a transcoder that produces a target bitstream with a lower bitrate than the original bitstream. This process of lowering the bitrate can be achieved by reducing the spatial resolution, reducing the frame rate, and/or increasing the quantization level of the coded video; in other words, bitrate reduction can be achieved by choosing an appropriate set of output coding parameters from the given input coding parameters. Many degrees of freedom exist when choosing appropriate source and target coding parameters; we guided our design decisions with the goal of creating a computationally simple transcoder that exploits the properties of the MPEG-2 and H.263 compression standards.

A good design requires an understanding of the performance loss due to recoding. Transcoding can be roughly viewed as a concatenated decode and re-encode operation; we call this concatenated operation a recoding cycle. When lossy compression techniques are used, recoding typically incurs additional quality losses in the coded bitstream, even when the same compression algorithm and bitrate are used for the input and output bitstreams. These losses stem from the fact that the video being re-encoded differs from the original video used in the initial encoder. Generally, larger differences between the original and initially coded video potentially result in larger recoding losses. A study was reported on the performance of MPEG-2 recoding cycles in studio (high bitrate) environments.[6] We are concerned with recoding performance when transcoding between the MPEG and H.263 standards for bitrate reduction and when working at lower bitrates.

Characterizing recoding performance has complications due to the fundamental difficulties associated with characterizing compression performance. For example, when compressing a given video sequence to a specified bitrate, there is no clear answer as to what is the best frame rate, spatial resolution, or quantization level. Similarly, when transcoding a given bitstream between specified source and target bitrates, there is no clear answer as to what are the best output coding parameters. However, in order to reduce the bitrate, it is clear that one or more of these parameters will need to be adjusted.

By conducting recoding experiments with various source and target bitrates, as outlined below, we determined that a practical approach would be to design a transcoder from 1.5 Mbps 30fps CIF MPEG to 500 kbps 10fps CIF H.263, using the MPEG and H.263 public domain source code[7,8] as a baseline. It was determined through experimentation that using an MPEG source bitstream at a rate less than or equal to 1.0 Mbps resulted in a far more significant degradation in quality when transcoded to H.263. Using these measurements as a benchmark, it was possible to develop various transcoders through modification of existing code, thus making comparison to original quality measurements more informative.

The first section discusses the results of recoding MPEG streams to H.263 streams at the *same* high bitrate, in order to determine recoding losses due to the different compression standards and independent of changing parameters. The second section focuses on selecting an appropriate target bitrate for the lower bitrate H.263 stream to minimize the recoding losses when transcoding for bitrate reduction.

MPEG $\rightarrow$ H.263 Transcoding with Similar Bitrates

In order to determine the relative success of our transcoding algorithm, it was necessary first to estimate a lower bound on recoding losses. Although MPEG and H.263 encoding are lossy, the minimum possible loss which we can hope to obtain from any recoding process can be estimated simply by recoding a high bitrate MPEG bitstream, i.e.

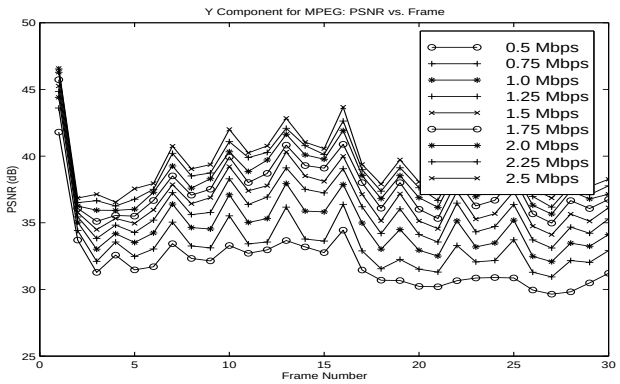

Figure 1. MPEG PSNR vs. Frame

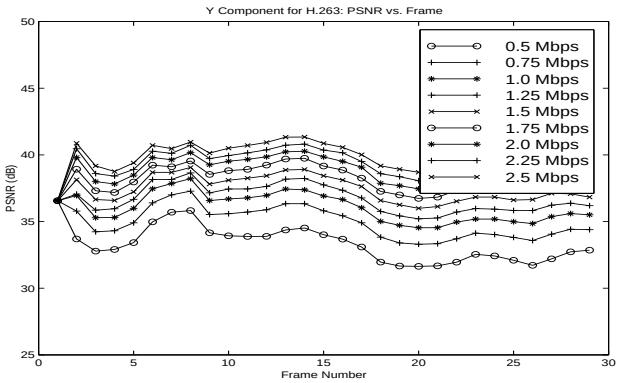

Figure 2. H.263 PSNR vs. Frame

by decoding the bitstream and re-encoding the decoded frames into an H.263 bitstream. Using these initial lower bounds as a baseline, it is possible to determine an appropriate source bitrate and to estimate the minimum possible loss which can be expected from a transcoding algorithm.

Figures 1 and 2 show the PSNR vs. frame number when encoding 30 fps CIF resolution 4:2:0 YUV data into MPEG and H.263 streams at bit rates ranging from 500 kbps to 2.5 Mbps. Figure 1 indicates that, for MPEG streams, the I and P frames have higher PSNRs than the B frames. This is to be expected, but is important when considering the mean PSNR vs. bit rate for MPEG streams, dropping B frames will result in a higher mean PSNR for the remaining frames. Additionally, when comparing mean PSNR for MPEG vs. H.263, it may be worthwhile to note that the variance of PSNR is much greater for MPEG than it is for H.263, due to its coding structure. Thus, while dropping B frames may result in an apparent higher mean PSNR for the sequence, there is the inherent tradeoff of accepting a lower frame rate.

When attempting to code the 30 fps CIF video sequences, it was also observed that, for bit rates below 500 kbps, the H.263 coder is unable to code at 30 fps and the MPEG coder is unable to code at all. At bit rates below 150 kbps, the H.263 coder becomes unable to code at 10 frames per second. The MPEG syntax simply does not support low frame rates and, as such, MPEG is unable to code at such low bit rates.

In figure 3 we plot the mean PSNR across the frames as a function of bit rate. The top and middle traces shows the mean PSNRs for the H.263 and MPEG streams. The bottom trace shows the mean PSNR that results when recoding the MPEG bitstream into an H.263 bitstream at the *same bit rate*. Notice that the recoded streams experience a 0.7-1.8 dB PSNR degradation from the original MPEG stream and about a 2 dB PSNR degradation from the original H.263 stream, when each are measured against the original YUV data. One goal of future work is to improve the quality of the recoded H.263 stream so that there is less quality loss in this estimated best-case scenario. This can be achieved by performing smart recoding rather than blind recoding. For example, the transcoder's re-encoder could perform better if it had knowledge of the quantization tables and motion vectors used in the original MPEG coder. These parameters could explicitly be passed to the transcoder's re-encoder or they could be derived from the decoded YUV frames.[9]

MPEG → H.263 Transcoding with Bitrate Reduction

Naturally, we are concerned with losses that result when recoding an MPEG bitstream into an H.263 bitstream at a *lower* target bitrate. The goal is to select an appropriate target bitrate which significantly reduces the bitrate without a substantial loss in quality. Furthermore, these experiments provide insight on lower bounds for quality losses which any reasonable algorithm could achieve.

Our transcoding algorithm first converts the 30 fps MPEG IPB stream into a 10 fps MPEG IP stream by dropping the B frames from the MPEG IPB stream. As mentioned earlier, we expect the mean PSNR of the remaining MPEG IP stream to be higher than that of the original MPEG IPB stream, which they indeed are. The 10 fps MPEG IP stream is then recoded into lower bit rate H.263 streams.

Figure 4 provides much insight about recoding lower bit rate H.263 streams from higher bit rate MPEG streams. Throughout this graph, the MPEG bit rates represent the total bitrate of the MPEG IPB stream, while the MPEG

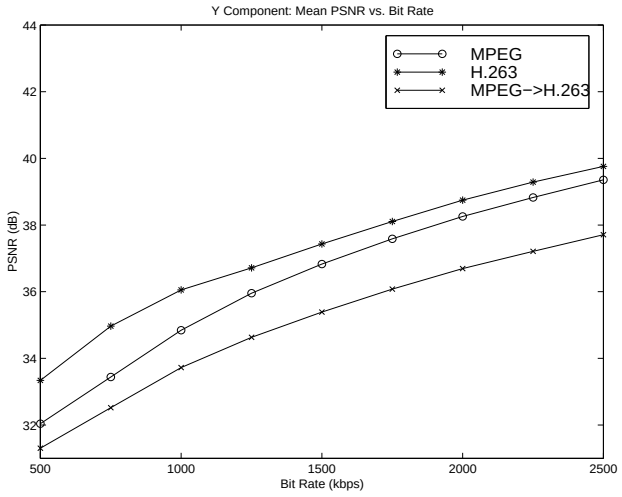

Figure 3. Mean PSNR vs. Bit Rate

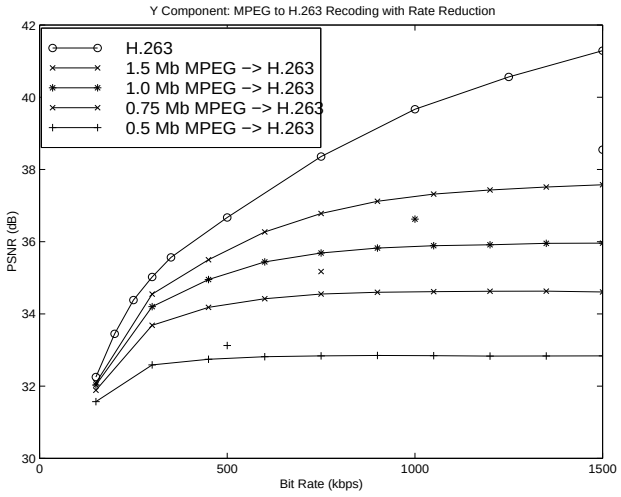

Figure 4. Recoding Losses

PSNRs represent the mean PSNRs of only the I and P frames. The H.263 traces represent the mean PSNRs and bit rates of the 10 fps H.263 recoded streams.

The individually plotted points on the graph represent the mean PSNR of the 10 fps MPEG IP stream at the appropriate MPEG IPB bit rate. These points provide upper bound values for their corresponding transcoded streams. For example, the PSNR of a 1.0 Mbps MPEG stream recoded into a low bit rate H.263 stream could never exceed the PSNR of this original MPEG stream. Likewise the PSNR of a recoded MPEG stream could never exceed that of the corresponding 10 fps H.263 stream encoded at the same bit rate.

From this plot we make the following observations: At a target bitrate of 150 kbps, all the MPEG sources result in target H.263 streams with mean PSNRs of about 32 dB. Thus, at very low target bit rates (on the order of 150 kbps), the bit rate of the original MPEG stream has less of an impact on the final quality of the encoded H.263 stream. Meanwhile, at higher target bitrates (those exceeding 500 kbps), higher quality MPEG sources (up to 1.5 Mbps) lead to higher quality transcoded streams. Specifically, at 500 kbps, using a 1.5 Mbps MPEG source is about 0.5 dB better than using a 1.0 Mbps MPEG source, and its performance is only about 0.5 dB away from the best case scenario of coding the original frames with the H.263 coder at 500 kbps. For this reason we chose to transcode CIF 30 fps 1.5 Mbps MPEG source bitstreams into CIF 10 fps 500 kbps H.263 target bitstreams.

3. INTER-STANDARD DIFFERENCES: PROBLEM DESCRIPTION

Developing an application which effectively provides an interface between two standards is always a challenge; when developing an application which operates between two different standards, the different approaches to problems taken by each standard can make interoperability difficult. A number of these differences exist between MPEG and H.263; these differences are outlined in the table below.

Differences between MPEG and H.263

	MPEG	**H.263**
Video Formats	Progressive and Interlaced	Progressive Only
I Frames	More (random access)	Fewer (compression)
Frame Coding Types	I,P,B Frames	I,P, Optionally PB
Prediction Modes	Field, Frame, 16 × 8	Frame Only
Motion Vectors	Inside Picture Only	Can point outside picture

With the given implementation of an MPEG decoder and H.263 encoder, an MPEG frame tends to contain far more intra coded macro blocks than does its corresponding H.263 frame. Furthermore, while H.263 is coded with I

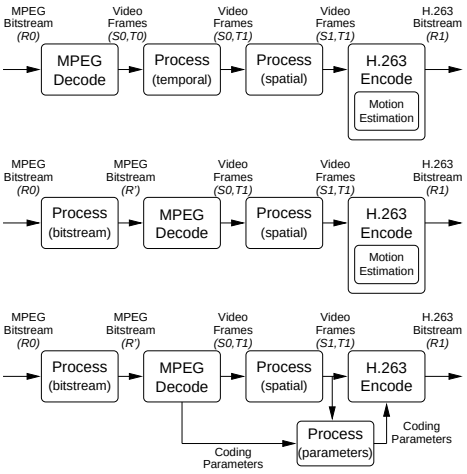

Figure 5. Development of Proposed Approach

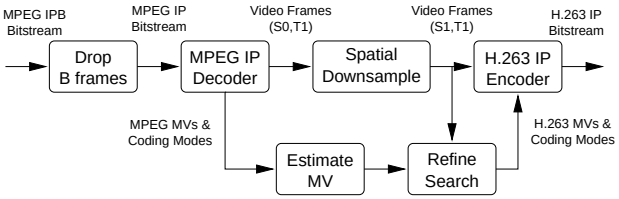

Figure 6. Detailed Block Diagram of Approach

(intra-coded) and P (forward-prediction) frames (and optionally one PB frame between a pair of I and P frames), MPEG uses I, P, and B frames, with an I frame for every GOP.

The H.263 standard allows for an unrestricted motion vector mode, which allows motion vectors to point outside the given frame, an attribute not shared with MPEG. As such, the transcoder can be expected to experience quality losses around boundaries, especially in sequences which involve movement in boundary neighborhoods.

While MPEG allows for interlaced video formats, H.263 merely allows for progressive formats; thus, any MPEG transcoding algorithm which accepts MPEG interlaced formats must devise a method for converting this to a progressive format.

The transcoding algorithm we present is forced to address these issues. First, it drops MPEG B frames before the decoding process, thus reducing the resulting stream to 10 fps. Second, the algorithm converts many of these MPEG intra macroblocks to inter blocks (especially in the case of a predictable sequence), using as motion vectors the vectors from the corresponding macroblock in the preceding P frame. Finally, we address field to frame conversion with various heuristics for estimating motion vectors, as well as spatial downsampling.

4. AN MPEG TO H.263 TRANSCODER

The transcoding algorithms described in the subsections below are steps in the evolution of our transcoding algorithm. After the appropriate source and target bitrates were determined, a logical step was to create a "conventional" transcoder by simply concatenating an MPEG decoder and an H.263 encoder, performing all processing in the pixel domain. The second method transcodes more efficiently with a minimal loss in quality by making use of motion vector information from the decoded MPEG stream; this method is subsequently enhanced to handle MPEG-2 CCIR input. The final algorithm combines motion vector reuse with a limited search to obtain the best motion vectors with which to code the H.263 sequence.

Figure 5 summarizes the development of the transcoding algorithm. The top diagram shows the conventional approach to transcoding, whereby all processing is performed in the frame domain, both temporally and spatially. The second diagram depicts the first enhancement, performing temporal downsampling in the MPEG bitstream domain. The third and final approach further enhances this method by using coding parameters from the decoded MPEG bitstream to bypass motion estimation, reducing both memory usage and CPU cycles. The following subsections outline this development in greater detail.

Algorithm Summary

The resulting algorithm downsamples temporally by a factor of 3 (dropping MPEG B frames). Given MPEG CCIR-601 interlaced input, the algorithm also downsamples spatially by a factor of 2 (by extracting the top field of the MPEG interlaced video frame and horizontally downsampling it by a factor of 2). Temporal downsampling reduces the bitrate dramatically while allowing motion vectors for the most important frames (MPEG P frames) to be reused.

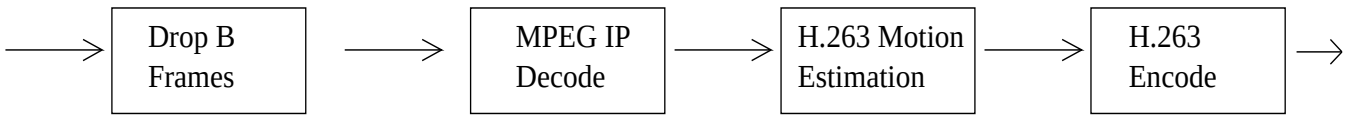

Figure 7. Method 1: The Conventional Approach

Spatial downsampling also reduces the target bitrate. Furthermore, vertical downsampling allows us to avoid the difficulties associated with interlaced to progressive conversions, and horizontal downsampling resolves some of the limitations imposed by the MPEG motion vector format.

Methods for obtaining motion vectors for the corresponding P frames in H.263 varied according to the method, as well as whether or not the input was interlaced. While the first method simply recalculates all motion vectors for the H.263 P frames, the second method uses the motion vectors from the corresponding MPEG P frame (or the most recent MPEG P frame if the current frame is an MPEG I frame). These copied motion vectors are then refined with a half-pel search. Figure 6 summarizes the resulting transcoding algorithm. This basic algorithm is subsequently enhanced to handle MPEG interlaced input.

Method 1: The Conventional Approach

This initial method, outlined in Figure 7, employs no heuristics for improving the efficiency of the transcoding process. The idea is to operate completely inside the pixel domain, reperforming all aspects of the MPEG decoding and H.263 encoding process. MPEG B frames are dropped in an effort to reduce the bitrate of the resulting H.263 stream.

There is an inherent tradeoff between the resolution and quality of the transcoded sequence and the computation required for the transcoding process. The object is then to develop an effective algorithm for speedy transcoding while preserving the quality of the resulting sequence. Using a higher bit rate for the source stream (MPEG) results in less quality loss, due to less loss in the recalculation of motion vectors. As one would expect, while recomputing motion vectors from the decoded MPEG frames may result in the highest quality picture, the computation required for this method is much too great to use for a real time transcoder. Although this method is an obvious one to try, as expected, it results in a transcoder which is much too sluggish for our needs. In order to create an efficient transcoder, we can look for information in the decoded MPEG stream to reuse in the re-encoding of the H.263 stream. As we shall see, making use of available information saves a considerable amount of time.

Method 2: Reusing Motion Vector Information

The second method, illustrated in Figure 8, reuses information from the MPEG bitstream, specifically motion vector information, in order to bypass the computationally intensive step of motion estimation. In order to account for some of the previously described shortcomings of MPEG motion estimation and to recover some of the potential loss in image quality, these copied motion vectors are refined by performing a half pel search in the horizontal and vertical directions; selection of the motion vector from these nine possibilities allows much more of the image quality to be preserved.

By using the motion vectors from the decoded MPEG stream and bypassing motion estimation when re-encoding the H.263 stream, the algorithm is able to make use of information which already exists about the motion of each macroblock, thereby speeding the process. As in Method 1, the algorithm takes MPEG CIF and converts this to H.263 CIF at a lower bitrate; no spatial downsampling is performed; the sequence is downsampled temporally by a factor of 3.

Many issues arise when these motion vectors are simply copied without regard to the differences between MPEG and H.263 motion vector fields. Finding the causes of these discrepancies in the motion vectors motivated exploration of many factors. H.263 allows the use of a feature known as unrestricted motion vector mode, which, among other things, allows motion vectors to point outside the frame itself. One can see that for particular sequences for which there is motion near or around boundaries, this shortcoming of MPEG motion vectors becomes increasingly apparent. One goal was to figure out exactly how much of the total quality degradation in terms of PSNR could be attributed to this error around the frame boundary. This error, although sequence specific, can be recovered mostly from other methods, such as the half pel search motion vector refinement described below.

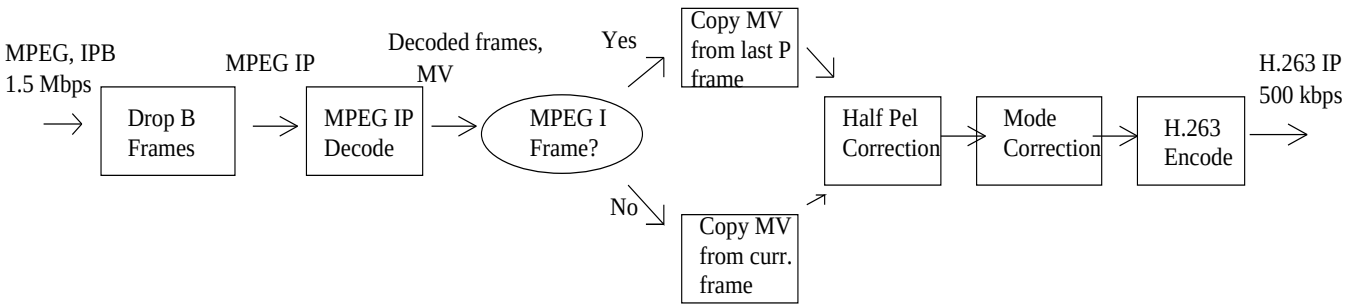

Figure 8. Method 2: Reuse Information From MPEG

I Frame to P Frame Conversion

Because MPEG provides less motion vector information than H.263 may expect to effectively code a particular frame, the algorithm must somehow include a method for determining motion vectors for macroblocks for which the MPEG stream fails to provide motion vectors. Because MPEG tends to code many more macroblocks as intra macroblocks than does H.263, quality of the H.263 frames could be improved by making many of these blocks inter macroblocks and supplying them with appropriate motion vectors.

After determining the benefits of converting MPEG intra macroblocks to inter blocks for the purposes of H.263 encoding, various techniques were attempted in order to adequately predict what these motion vectors might be. These methods included using the corresponding motion vector from the preceding P frame and taking a weighted average of neighborhood motion vectors in the preceding P frame for some or all of the MPEG intra macroblocks. It was determined that the most effective method of converting these MPEG intra macroblocks was to use the corresponding motion vector from the preceding P frame to predict the motion vector, and subsequently use the mode selection method as defined in the H.263 encoder to determine the appropriate macroblock mode; if the resulting error was above a certain threshold, the macroblock remained intra.

Motion Vector Refinement

By observing the errors which resulted from the motion vectors with respect to those computed by the original H.263 encoder (Figure 15), it was evident that a half-pel search from the MPEG motion vectors in order to minimize error could greatly increase the quality of the transcoded frame without noticeably slowing the algorithm. Thus, we use the information provided to us from the MPEG motion vectors to conduct a heuristic search to find a motion vector for the H.263 encoding process which minimizes error for a given search area. Of course, one would expect to see slight quality improvements in exchange for a search time which grows as n^2, where n is the number of pixels to search in each direction. Because of this relatively unfriendly order of growth and our focus on efficiency, it is in our best interest to limit the scope of this search while still gaining some error resilience; considering such factors, a half pel search turns out to be both efficient and beneficial in terms of error reduction.

Method 2 Enhanced: MPEG Interlace as Input

The method used to transcode interlaced input is largely similar, with a few important distinctions. First of all, the frames of the interlaced input have a spatial resolution of 720 × 480, so we must first downsample the decoded MPEG frames to our prior spatial resolution of 352 × 240; to obtain 352 pixels, we sample alternate pixels to obtain 360 and subsequently crop the rightmost 8 columns. The U and V components are subsampled from 360 × 240 to 176 × 120. To do this, we sample from the top field of the original picture, averaging neighboring pixels. Next, the motion vectors from the decoded MPEG fields are preserved, but only the motion vectors from the top frame of the sequence are utilized, according to the resulting picture which we have downsampled.

I Field to P Frame Conversion

Several complications arise due to the motion vectors which are available to a particular field. For each field, various choices are available for the motion vector. If a field is the top field of frame, each macroblock has two choices for its motion vector: one which is derived from the top field of the previous frame and one which is derived from the bottom field of the previous frame. If it is the bottom field of the frame, each macroblock has two options for its

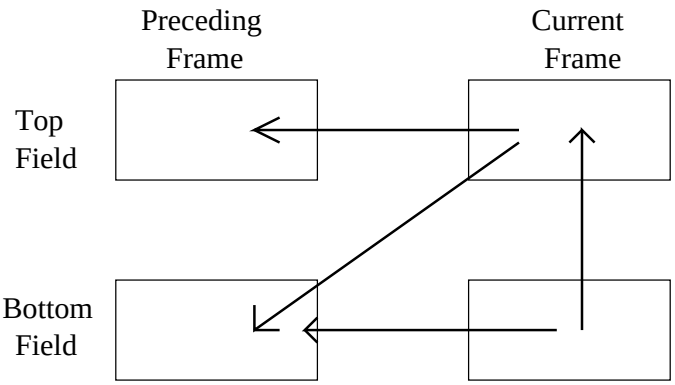

Figure 9. Motion Vector Dependencies for Field Mode

motion vector: one which corresponds to the top field of the same frame, the other to the bottom field of the previous frame. These dependencies are depicted in Figure 9. Although each macroblock has two motion vectors associated with it, it only uses one of these during motion compensation. As in the previously described method, in the case of an MPEG I frame we use the motion vectors from the preceding P frame.

P Field to P Frame Conversion

Because the algorithm samples the top field only, the algorithm attempts to obtain motion vector values which correspond to the top field of the preceding frame as often as possible. Since we are downsampling horizontally as well, if the particular macroblock at which we are looking does not use the motion vector which corresponds to the top field of the preceding frame, we can check to see if the neighboring macroblock has a motion vector which corresponds to the top field of the preceding frame. Surely this is more effective than using a motion vector which corresponds to the bottom field, as we are considering images drawn from the top field only.

Additionally, in an interlaced sequence, the MPEG encoder has the option of coding two motion vectors for each 16×8 square in any given macroblock. Each one of these motion vectors can correspond to either preceding field, as described above. If a block is coded as two 16×8 blocks, the algorithm checks both of these blocks to try to find a motion vector which corresponds to the appropriate frame. This can be done for both the current macroblock, and, if failure ensues, for the neighboring macroblock so that, best case, there are four motion vectors to choose from. If more than one of these corresponds to the top field of the preceding frame, the selection order is arbitrarily defined to be left to right, top to bottom. Only about 10% of all macroblock pairs (i.e., a macroblock and its right neighbor) fail to have any motion vectors which correspond to the appropriate field. In this case, we pick the upper right motion vector, which corresponds to the bottom field of the previous frame.

Once an acceptable set of motion vectors for our macroblock subset is obtained, the algorithm proceeds to check each half-pixel neighborhood of each motion vector in an attempt to minimize error further, as before. Following this, a similar mode correction is performed.

5. OBSERVATIONS AND RESULTS

In this section, the experimental results are discussed for the various transcoding methods. The reuse of motion vectors results in a $9-10\times$ speedup for transcoding MPEG CIF progressive 30 fps to H.263 CIF progressive 10 fps, and a $7-8\times$ speedup for transcoding from MPEG CCIR interlaced 30 fps to H.263 CIF progressive 10 fps.

It is observed that the reuse of motion vectors from the decoded MPEG bitstream results in a significant speedup from the conventional algorithm without a significant loss in quality. The algorithm also successfully transcodes interlaced input to H.263 progressive without a significant quality loss by reusing various field motion vector information and utilizing spatial and temporal downsampling.

The nature of interlaced video (i.e., two fields per frame, alternating lines), allows for ease in vertical spatial downsampling. Furthermore, spatial downsampling in the horizontal direction combined with the reuse of the original motion vectors permits more flexibility in motion vector selection, thus minimizing the initial loss in quality.

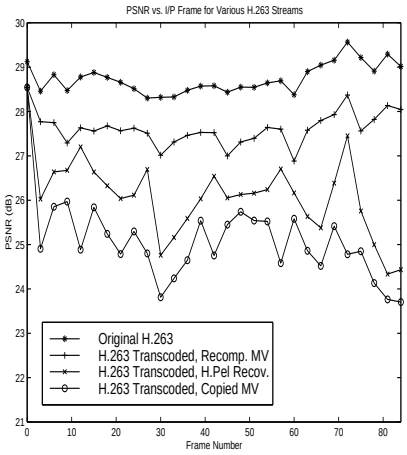

Figure 10. Bus Sequence

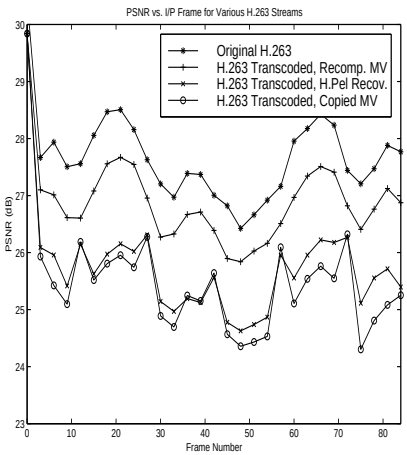

Figure 11. Carousel Sequence

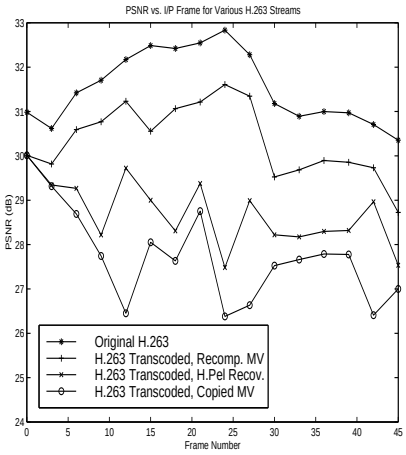

Figure 12. Girl Sequence

Recalculating vs. Reusing Motion Vectors

The table below depicts the savings in computation time resulting from using Method 2, which employs the motion vectors from the decoded MPEG bitstream. Additionally, note the time which is gained is particularly helpful when the MPEG bitstream contains motion vectors which have horizontal or vertical component greater than 16; these vectors, which would ordinarily have to be discovered through a full spiral search conducted by the H.263 motion estimation module, can simply be passed to the H.263 encoder, thus saving a considerable amount of search time. All values shown below make use of the H.263 encoder's unrestricted motion vector mode. The results show about a 10× efficiency improvement for an average of about 1.5 dB quality loss for this particular implementation of the H.263 encoder. This is in addition to the approximate 0.5 dB loss which is incurred by simply decoding the MPEG and re-encoding H.263 images at the same bit rate (this will be greater, since we are recoding at a lower target bit rate).

Transcoding Times for Various Sequences (I/P Frames) and Algorithms

Sequence	*Method 1 Time (sec)*	*Method 2 Time (sec)*
Carousel (30 IP)	323	26
Bus (30 IP)	254	26
Girl (18 IP)	106	15
Bus (10 IP)	80	9
Carousel (10 IP)	90	8
Football (10 IP)	92	8

Below are various PSNR measurements for a some sequences. There are many interesting observations we can draw from these measurements.

Figures 10, 11, and 12 indicate that the nature of the sequence has a direct bearing on the benefit which recomputing half pel motion vectors provides. The bus sequence contained primarily translational motion in one direction. Thus, small errors in motion vectors account for much of the PSNR loss; the benefits of a half pel search are evident. In the carousel sequence, however, the benefits of a half pel search are less pronounced. This suggests that motion vector error as decoded from this MPEG sequence consists of errors which are of a different character.

A half pel motion vector search can be useful in sequences such as the bus sequence, where much of the PSNR loss is due to small errors in motion vector values. For example, in a particular frame of the bus sequence, we can see the copied motion vectors in Figure 13 and the recalculated motion vectors in Figure 14 and the corresponding error vectors in Figure 15. Clearly, much of the quality loss in this particular frame, and generally in this sequence, is due to small nonzero errors in the motion vectors. These types of errors can be alleviated by half pel search.

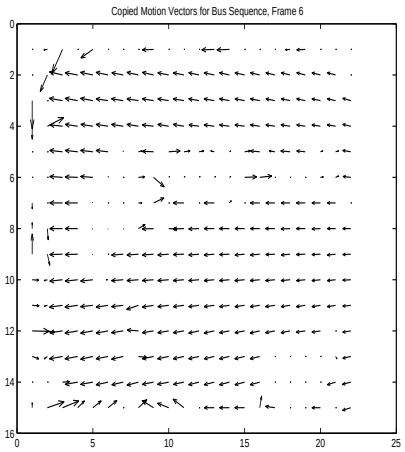

Figure 13. Copied MV

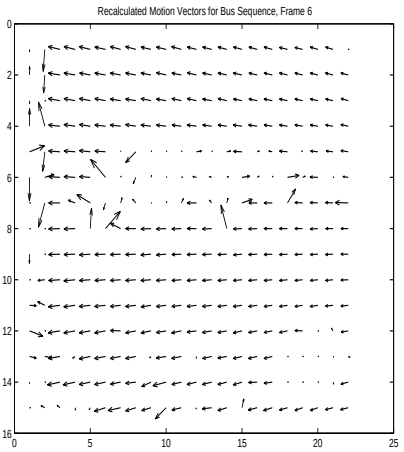

Figure 14. Recalculated MV

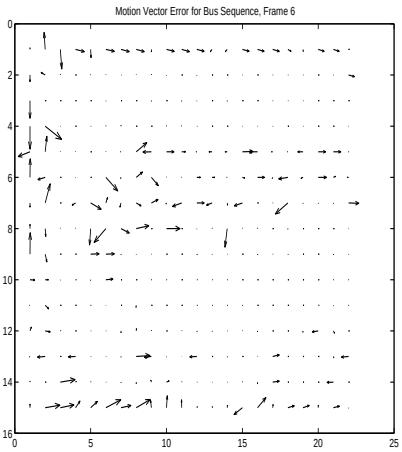

Figure 15. Error

Additionally, it is evident that many of the discrepancies between the two motion vector arrays results in boundary calculations.

Another noticeable difference between the sequences is the impact of MPEG I frames on the PSNR of the corresponding transcoded frame. According to the rate control method specified, I frames were allotted 125% of the amount of memory given to P frames (50K vs. 40K), and the initial I frame in a sequence tends to take more bits than subsequent I frames. We are concerned with generating a high quality I frame, because, as can be observed in the above figures, a good I frame results in less degradation in future predictions in the sequence.

In a sequence where the motion vectors correspond highly to the nature of the motion in the picture and are redundant, coding the MPEG I frames as I frames (as would occur when motion vectors are copied directly) results in a noticeable PSNR degradation. This can be seen, for example, at frame 12 in Figure 10. By copying the motion vectors from the MPEG sequence directly (thus coding the first picture in each GOP as an I frame), useful predictive information about the picture is discarded. Thus, in this case, our method which employs copying motion vectors from the preceding MPEG frame results in a great improvement.

On the other hand, in a sequence where the motion vectors are less redundant, such as the carousel sequence, copying the motion vectors from the previous P frame can result in a performance that is slightly worse than that of simply coding the frame intra. This is because the predictive value of the preceding frame's motion vectors is much less, due to decreased redundancy. When the mode selection module is employed, this principle exhibits itself: given a sequence with highly redundant motion vectors (i.e., a predictable sequence), more macroblocks will tend to be inter coded; otherwise, more macroblocks will be intra coded, as the prior motion vector information does not provide as much predictive value. Generally, given an MPEG I frame in a predictable sequence, copying motion vectors from a prior P frame yields more favorable results than intra coding (and for this particular implementation, also saves bits).

Results for MPEG Interlaced Input

By downsampling the image horizontally, we can avoid using motion vectors from the macroblocks located along the right boundary (Figures 16 and 17). This results in a solution to the boundary problem which resulted previously from the limitations of the MPEG standard. Additionally, downsampling vertically allows us one pixel of leeway along the bottom boundary. It is also worthwhile noting that processing interlaced 720 $\times$ 480 input increased transcoding time for 10 IP frames to 12 seconds, compared with 8 seconds for 352 $\times$ 240 progressive input.

Transcoding Times for Interlaced Sequences and Algorithms

Sequence	*Method 1 Time (sec)*	*Method 2 Time (sec)*	*Mean Loss (dB)*
Bus (10 IP)	80	12	1.58
Carousel (10 IP)	100	12	1.19

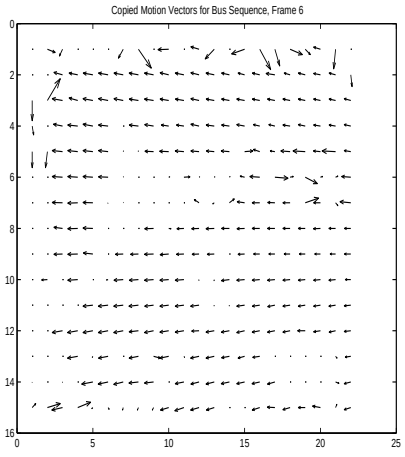

Figure 16. Copied MV

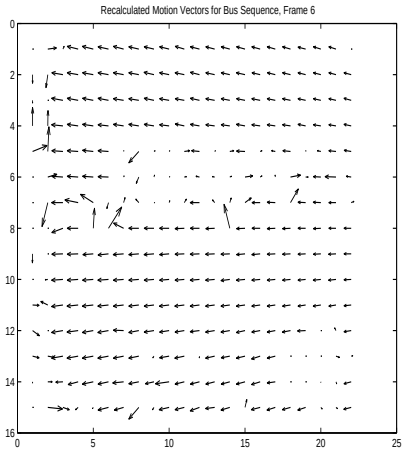

Figure 17. Recalculated MV

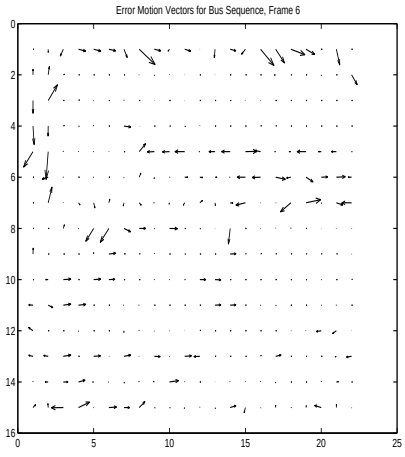

Figure 18. Error

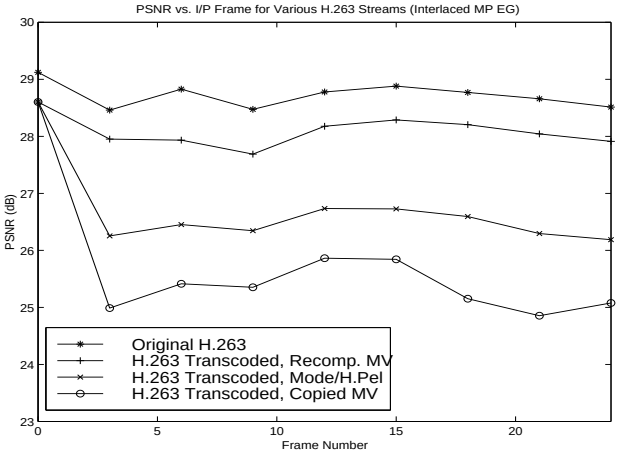

Figure 19. Bus, Interlaced Input

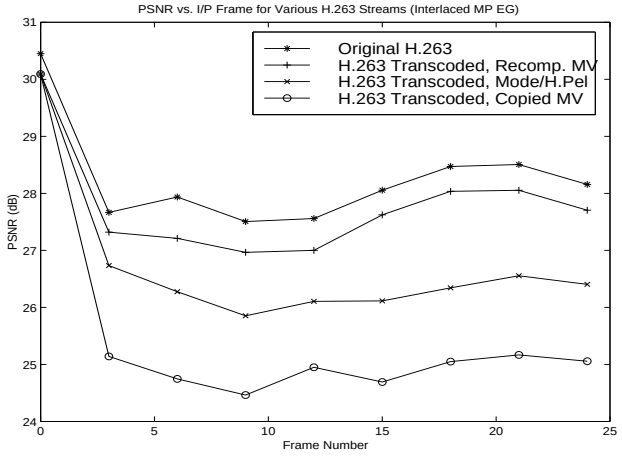

Figure 20. Carousel, Interlaced Input

Choosing the appropriate motion vector from the four available choices (i.e., one that corresponds to the top field of the previous frame) can result in a gain of approximately 0.5 dB over simply picking the top left motion vector every time. Since there exist at least two and at most four possibilities for motion vectors to select, on 90% of all occasions it is possible to find a motion vector which corresponds to the appropriate field. Half-pel motion vector refinement and mode correction described previously provided a benefit of approximately 1.1 dB in the case of interlaced input. The resulting picture still experiences a loss of approximately 1.4 dB on average.

6. CONCLUSIONS AND FURTHER RESEARCH

By using a transcoding algorithm which makes use of motion vector information from the MPEG bitstream, transcoding efficiency can be significantly improved. For this particular implementation, reusing motion vector information saved 85-90% of the computation time required without such a method. Additionally, the resulting picture experienced a minimal degradation in quality, approximately 1.4 dB for both progressive CIF and CCIR-601 MPEG source streams. This loss can be explained partially by quantization losses and also by boundary errors.

There still remain many areas for potential improvement of the MPEG $\rightarrow$ H.263 transcoding algorithm. In order to enable a real time transcoder, the algorithm must be made more efficient. Currently, the transcoder operates at about 1 frame per second. While this is considerably faster than previously, efficiency can still be improved with algorithmic and implementation optimizations. One might also want to experiment with different transcoder outputs, such as lower spatial resolution and lower target bitrates.

Another interesting project could be the investigation of an effective way to estimate boundary motion vectors, as MPEG does not provide a specification for motion vectors which point outside of the frame. Along the lines of motion estimation, one could also investigate better ways to predict the motion vectors for macroblocks in MPEG I frames, as well as more effective methods of determining the appropriate coding mode for each macroblock. Finding a method which accurately predicts motion vectors for MPEG I frames consistently would prove to be a great benefit, as a high quality I frame can reduce degradation of subsequent P frames in the sequence. It would be best to look for a better means of finding these motion vectors which does not involve a weighted average of prior motion vectors. Along these lines, it would be best to avoid trying motion estimation for MPEG intra macroblocks, as this tends to result in minimal PSNR gain and is computationally expensive.

REFERENCES

1. H. Sun, W. Kwok, and J. Zdepski, "Architectures for MPEG compressed bitstream scaling," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 6, April 1996.
2. G. Keesman, R. Hellinghuizen, F. Hoeksema, and G. Heideman, "Transcoding MPEG bitstreams," *Signal Processing: Image Communication*, vol. 8, September 1996.
3. N. Bjork and C. Christopoulos, "Transcoder architectures for video coding," in *IEEE International Conference on Acoustics, Speech, and Signal Processing*, (Seattle, WA), May 1998.
4. T. Shanableh and M. Ghanbari, "Heterogeneous video transcoding MPEG:1,2 to H.263," in *International Packet Video Workshop*, (New York City, NY), April 1999.
5. S. Wee, J. Apostolopoulos, and N. Feamster, "Field-to-frame transcoding with spatial and temporal downsampling," in *IEEE International Conference on Image Processing*, (Kobe, Japan), October 1999.
6. C. Horne, T. Naveen, A. Tabatabai, R. O. Eifrig, and A. Luthra, "Study of the characteristics of the MPEG2 4:2:2 profile–application of MPEG2 in studio environment," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 6, June 1996.
7. MPEG Software Simulation Group, http://www.mpeg.org/MPEG/MSSG, *MPEG-2 Video Codec*, 1.2 ed.
8. University of British Columbia, http://spmg.ece.ubc.ca/h263plus, *H.263+ Public-Domain Code*, 3.2 ed.
9. Y. Chen, K. Challapali, and M. Balakrishnan, "Extracting coding parameters from pre-coded MPEG-2 video," in *IEEE International Conference on Image Processing*, (Chicago, IL), October 1998.