Improving Scene Cut Quality for Real-Time Video Decoding
Giovanni Motta\textsuperscript{1}, James A. Storer\textsuperscript{1}, and Bruno Carpentieri\textsuperscript{2}

\textsuperscript{1}Computer Science Department, Brandeis University
\textsuperscript{2}Dip. di Informatica ed Applicazioni "R.M. Capocelli", University of Salerno

Abstract: We address the problem of improving the scene cut quality in fixed bit-rate real-time video decoding such as is used in the H.263 and MPEG standards. In low bandwidth applications, scene cuts can cause the bits required to encode a single frame to greatly exceed the target average bits per frame, and necessitate the skipping of other frames to provide sufficient time to transmit the scene cut frame. We present an optimal algorithm for minimizing the number of skipped frames and keep the decoding synchronized. Although the algorithm requires additional encoding complexity, there is no change in decoding complexity (in fact, no change to the decoder at all). Experimental results, obtained with a simplified strategy in the framework of H.263+ video encoding, confirm that the method provides an effective alternative to current frame skipping strategies. The overall quality in the presence of scene cuts is improved with respect the TMN-8 rate control. Although the overall bit rate benefits from our method, our focus is to improve the quality of the video where scene cuts occur (by reducing skipped frames and improving decoder synchronization). The approach here can be combined with more sophisticated rate controls, like, for example, the newer Rate-Distortion optimized TMN-10 and TMN-11.

1. Introduction

Because of the high variability present in many video sequences, some frames, such as at a scene cut, may have an encoded size that is substantially higher than the average. If the encoder tries to achieve a quasi-constant bit-rate, for example to fully utilize a transmission channel, frames that have a large encoded size present a problem. Transmission of large frames may require a time substantially higher than the time available for a single frame, so in order to keep the sequence synchronized, the encoder is forced to skip the encoding of a few subsequent frames. The increase in rate at scene cuts is illustrated in Figure 1, which shows how the rate varies in 900 frames of one of the test sequences used in our experiments.

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{figure1.png}
\caption{Bit rate for Frames 1 to 900 of the sequence "Std100.qcif". The frames in this sequence are numbered starting at 0; Frame 0, which is not shown, goes to about 17,500.}
\end{figure}
Current video compression systems such as H.263 and MPEG1/2 use a hybrid coding scheme that combines motion-compensation and Discrete Cosine transform coding. Each video frame is partitioned in several macroblocks and each macroblock is encoded as a motion compensated difference with a previously sent macroblock (Inter frame coding or P-mode) or by direct quantization of its Discrete Cosine Transform coefficients (Intra coding or I-mode). In general, the encoder decides if a macroblock should be Inter or Intra encoded with by using empirical methods. A widely used one is to compare the absolute error of the prediction with a fixed threshold. We define a scene cut a frame in which the prediction model essentially fails (and a large number of I macroblocks is sent to the decoder) or when the scene contains too much movement.

Optimization of issues related to frame type selection and rate control have been considered by a number of others; for example: Kozen, Minsky, and Smith [1998] consider a linear programming approach to optimal frame dropping where a fixed number of frames are dropped in order to minimize the interval of non-playable frames in the MPEG encoded sequence. Wiegand, Lightstone, and Mukherjee [1995] present a dynamic programming algorithm to joint optimize frame type selection and quantization step in the framework of the H.263 video coding, where optimization is performed on each macroblock on a frame-by-frame basis; their experimental results show consistent improvement upon existing methods. Lee and Dickinson [1994] address a similar problem in the framework of MPEG encoding, where each group of frames is isolated and both frame type selection and quantization steps are optimized with a mix of dynamic programming and Lagrange optimization; unfortunately, their experiments show very little improvement (a possible reason may be the simplified approach that restricts the possible quantization steps to a very small set of values). Wiegand and Andrews [1998] present an approach based on rate-distortion that constitutes the base for the H.263+ Test Model Near-Term Version 10 (TMN-10 in short); the optimization is based on Lagrange multipliers, and experiments show that it is possible to set the multiplier to a constant value and keep the computational complexity low without sacrificing the performance.

2. The H.263 standard with TMN-8 rate control

Although our techniques apply to many video compression systems, including ones based on the MPEG standards, we will describe our results in terms of the H.263+ standard, since it is intended specifically for low-bandwidth real-time applications.

At the present, H.263+ (or H.263 Version 2) is regarded as state of the art low bit rate video coding and most of it constitutes the core of the incoming MPEG-4 standard. The encoder is optimized for low bit rate and for transmission on error prone channels. A number of options (negotiable options) can be used to increase quality, decrease bit rate or improve error resilience. Option should be negotiated between encoder and decoder.

Although the encoder is not standardized (H.263 is a decoder or bitstream standard) the committee proposes a Test Model Near-Term (TMN in short) for bit rate control and frame type selection. The current test model is the TMN-11. Even if we experimented with an encoder that uses the TMN-8 (because it is the one implemented in the publicly available encoder), our method can be combined with the TMN-10 or with the TMN-11 as well. For a detailed description of the test models, see Gardos [1997], [1998] and Wenger et al. [1999].
3. Problem Description

Figure 2 depicts the behavior of a H.263+ encoder when using the rate control strategy TMN-8 described in Gardos [1997].

| Frame No. | n-5 | n-4 | n-3 | n-2 | n-1 | n | n+1 | n+2 | n+3 | n+4 | n+5 | n+6 | n+7 |...
|-----------|-----|-----|-----|-----|-----|---|-----|-----|-----|-----|-----|-----|-----|----|
| Encode    | n-5 | n-4 | n-3 | n-2 | n-1 | n | skip | n+4 | skip | n+6 | n+7 |...
| Transmit  | n-5 | n-4 | n-3 | n-2 | n-1 | n | n+4 | n+6 | n+7 |...
| Display   | ... | n-5 | n-4 | n-3 | n-2 | n-1 | n | n+4 | n+6 | n+7 |...

**Figure 2:** Encoding, transmitting and decoding/displaying a sequence of frames with a H.263 encoder using TMN-8 rate control. The sequence contains a scene cut between the frames n-1 and n.

After the first frame of the sequence is encoded in Intra mode (all macroblocks are Intra), almost every other frame is encoded by using the motion compensated prediction (Inter mode). If we focus our attention around a scene cut (frame n in the Figure 2) we note that this frame takes in general a considerable time to be transmitted (in this case, 3 additional frames of time). In the meanwhile, while waiting for the complete reception of the frame n, the decoder keeps showing the frame n-1 on the screen. To maintain synchronization with the original sequence, in this example the encoder is forced to skip 3 frames during the transmission of frame n and because of this skipping, the next frame to be encoded will be frame n+4.

For frame n of Figure 2, it was necessary to skip 3 frames. In general, as depicted in Figure 3, there will be some number k≥0 such that:

1. There are k extra units of time in which the frame n-1 is frozen on the screen (k=3 in Figure 2).
2. There is a "jerk" between the frame n and the frame n+k+1.
3. Because the frames n and n+k+1 are not contiguous in time and n+k+1 is predicted from n, the prediction error frequently generates a frame n+k+1 that is too big to be sent in one unit of time (n+4 in the Figure 2). This forces the encoder to skip also the frame n+k+2 before encoding n+k+3.

We consider reduction of the effects described in Items 2 and 3 through the optimal selection of the first frame that should be encoded (frame n in Figure 2) immediately after the termination of the previous scene (frame n-1 in Figure 2). This is of course only possible if the encoder has some look-ahead capabilities (and it may be that encoding is not done in real-time). However, decoding is unaffected (the encoder is not aware that this optimization has taken place), which is all that matters for many practical applications such as video distribution where a powerful encoder creates a compressed video sequence that is distributed in real time to many decoders.
Figure 3: Sequence of 50 frames across a scene cut in one of the files used in our experiments (Claire.qcif 80-100 followed by Carphone.qcif 1-29) encoded at 32 Kbit/s with TMN-8 rate control; the bits per frame and the corresponding PSNR per frame are shown.

4. An Optimal Algorithm Based on Dynamic Programming

In this section we present a dynamic programming algorithm that produces an optimal encoding under one very reasonable assumption. We consider an encoding of a sequence of frames at a given quality optimal if it skips as few frames as possible and among encodings that skip the same number of frames, uses as few bits as possible. We employ the following notation:

\[ F[1]..F[n] = \] A sequence of frames.

\[ r = \] The number of frames per second.

\[ B = \] The number of bits per frame in the uncompressed video.

\[ M = \] The maximum number of bits that can be sent in 1/r seconds.

\[ q = \] The minimum quality of a frame; e.g., maximum mean squared error between a compressed frame and the original one.

\[ d = \] The maximum possible number of sequential frames that can be skipped.

For any value of \( q, d \) is a constant independent of \( n \) that is at most \( B/M \);
that is, at most the number of bits to send the frame with no loss divided by the number of bits that can be sent in \( I/r \) seconds (or perhaps \( d \) is equal to \( B/M \) plus a few extra bits for header information).

\[
R[i,j] = \text{The maximum possible residual capacity (number of unused bits) when } F[i]...F[n] \text{ is encoded under the assumption that the first frame that is not skipped is predicted from } F[i-j], 1 \leq i \leq n, 0 \leq j \leq d.
\]

**Note:** To make \( R[i,j] \) well defined, we assume that frames \( F[-d+1]...F[0] \) are frames that are no help in predicting any frames of \( F[1]..F[n] \).

\[
S[i,j] = \text{The number of skipped frames in the encoded sequence corresponding to } R[i,j], 1 \leq i \leq n, 0 \leq j \leq d.
\]

\[
L[i] = \text{The index of the first non-skipped frame in the best of the encoded sequences corresponding to one of } R[i,j], 0 \leq j \leq d; \text{ that is, } L \text{ store the links that tell us, starting with position } i, \text{ which positions should be encoded.}
\]

We make the following assumption:

**Assumption 1:** There is a cost \( P[i,j] \) of predicting \( F[i] \) from \( F[i-j] \) compressed at quality \( q \) that is independent of how \( F[i-j] \) was compressed (so long as the quality is \( q \)), \( 1 \leq i \leq n, 0 \leq j \leq d. P[i,0] \) is simply the cost of encoding \( F[i] \) as an I frame.

**Note:** Like \( R[i,j] \), to make \( P[i,j] \) well defined, we assume that frames \( F[-d+1]...F[0] \) are frames that are no help in predicting any frames of \( F[1]..F[n] \).

Although Assumption 1 is not likely to be precisely true in practice, it is likely to be a very good approximation to what happens. That is, two encodings of a frame at the same quality are likely to be equivalent in their ability to predict a subsequent frame.

We also make the following second assumption to simplify our presentation; it is not necessary, and at the end of this section we note how the algorithm can be modified (without significantly changing the time or space complexity) to not require it.

**Assumption 2:** \( P[i,j] \leq P[i,j+1] \leq P[i,0], 1 \leq j < d. \)

Referring to the dynamic programming algorithm presented on the next page, we proceed from right to left in \( F \) in a fashion that can be thought of as a generalization of optimal text paragraphing. The outer for loop progress from right to left through \( F \). Iterations of the second for loop can be done in any order (or in parallel); it begins by initializing \( R[i,j] \) and \( S[i,j] \) to degenerate values and making \( k_{max} \) less than \( d \) when \( i \) is close to \( n \). The third for loop now tries each of the positions to place the first non-skipped frame in a best possible encoding of \( F[i]..F[n] \) (one with the minimum number of skipped frames and among encodings with a minimum number of skipped frames, the one with the most residual bits). The first if statement (Statement 1) calculates the number of frames worth of data that it will take to send frame \( i+k \) when predicted from the frame \( i-j \) (or as an I frame in the case \( j=0 \)). The second if statement (Statement 2) checks whether \( i \) is so close to \( n \) that there are not enough frames left to accommodate the sending of \( F[i+k] \), and in this case skips the remaining frames, assigned the appropriate credit to \( R[i,j] \), and sets \( L[i] \) to a degenerate value. The else statement (Statement 3) calculates the residual from sending \( F[i+k] \), calculates the associated number of skips, and then updates \( R, S, \) and \( L \) if a better encoding has been found. Due to assumption 2, \( L \) needs to be updated only when \( j=0 \) or \( j=1 \).
Dynamic Programming Algorithm for Optimal Frame Skipping:

A. for $1 \leq i \leq n$ do begin
   Compute $P[i, 0]$ with an I-frame encoding.
end

B. for $1 \leq i \leq n$ and $1 \leq j \leq d$ do begin
   Compute $P[i, j]$ based on a prediction from the I-frame encoding of $F[i, j]$ to
   achieve quality $q$, or define $P[i, j] = P[i, 0]$ if $i > 1$.
end

C. for $i := n$ downto $1$ do begin
   for $0 \leq j \leq d$ do begin
      $R[i, j] := 0$
      $S[i, j] := \infty$
      for $0 \leq k \leq \text{MINIMUM}(d, n-i)$ do begin
         1. Compute the number of frames worth of bandwidth needed:
            if $j = 0$
               then $x := \lfloor P[i+k, 0]/M \rfloor$
               else $x := \lfloor P[i+k, j+k]/M \rfloor$
            2. Check if there is not enough bandwidth left for the frame:
               if $(i+x-1) > n$ then begin
                  $S[i, j] := n-i$
                  $R[i, j] := S[i, j]*M$
                  $L[i] := n+1$
               end
            3. Compute the residual and number of skips used and check if we are
               better off than what we have already:
               else if $k \leq x$ then begin
                  $\text{residual} := x*M - P[i+k, j+k] + R[i+x, x-k]$
                  $\text{skips} := S[i+x, x-k] + x - 1$
                  if $\text{skips} < S[i, j]$ or ($\text{skips} = S[i, j]$ and $\text{residual} > R[i, j]$) then begin
                     $R[i, j] := \text{residual}$
                     $S[i, j] := \text{skips}$
                     if $i > 1$ and $j \leq 1$ then $L[i-1] := i+x-k+1$
                  end
               end
      end
   end
end

The dynamic programming of Step C is $O(d^2 n)$, which is $O(n)$ since $d$ is a constant
independent of $n$. Since $n$ is simply the number of frames, which is far less than the total
number of bits in all frames, in practice the time for Step C is likely to be insignificant
compared to Steps A and B, which essentially encode each frame $d+1$ times. For
applications where encoding is done only once and decoding is done many times (for
example, video servers or placement of compressed video on DVD), increasing the
encoding time by a factor of $d$ (e.g., $d=6$ in practice) may not be a significant problem.

Although it models most practical systems, Assumption 2 can be avoided by replacing
the final if statement by code that checks all potential positions $j$; since there are at most $d$
positions, the $O(n)$ running time is not changed. The algorithm can also be modified to
allow predictions from distances greater than $d$, although the time may be increased.
5. Experimental Results

To assess the performance of the proposed method, we embedded the frame selection strategy inside a publicly available H.263+ encoder (www.ece.ubc.ca/image). Three test sequences were generated so that they would contain a reasonably large number of scene cuts. The sequence std.qcif consists of the simple concatenation of the 9 files in a widely used standard data sets (claire.qcif, carphone.qcif, foreman.qcif, grandma.qcif, miss_am.qcif, mthr_dotr.qcif, salesman.qcif, suzie.qcif and trevor.qcif). The sequence std100.qcif was built interleaving the files in the standard data sets, each of them taken in blocks of 100 frames. The sequence commercials.qcif is a sequence of TV commercials. Figure 4 on the top shows sample frames from both data sets.

![Sample Frames](image1)

![Sample Frames](image2)

**Figure 4:** The top strip shows a sample frame from each of the files forming std.qcif and std100.qcif; the bottom strips show sample frames from each of the commercials forming commercials.qcif.

We have experimented with a simple (but not necessarily optimal) fixed strategy of selecting the frame that ends the sequence of skipped frames in the TMN-8 rate controlled encoder. This has the advantage of minimizing both the jerk and the distance to the next encoded frame. Although the optimal algorithm presented earlier is linear time, this approach reduces the computation to essentially at most two encodings of every frame. The Figure 5 shows how this simple strategy works on the same sequence of frames described in the Figure 2. As can be seen from Figure 5, the frame $n+3$ is encoded instead of the frame $n$ and because $n+4$ is predicted from $n+3$, no additional skipping is in general necessary.

| Frame No. | n-5 | n-4 | n-3 | n-2 | n-1 | n | n+1 | n+2 | n+3 | n+4 | n+5 | n+6 | n+7 | ...
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Encode</td>
<td>n-5</td>
<td>n-4</td>
<td>n-3</td>
<td>n-2</td>
<td>n-1</td>
<td></td>
<td>skip</td>
<td></td>
<td>n+3</td>
<td>n+4</td>
<td>n+5</td>
<td>n+6</td>
<td>n+7</td>
<td></td>
</tr>
<tr>
<td>Transmit</td>
<td>n-5</td>
<td>n-4</td>
<td>n-3</td>
<td>n-2</td>
<td>n-1</td>
<td></td>
<td>n+3</td>
<td></td>
<td>n+4</td>
<td>n+5</td>
<td>n+6</td>
<td>n+7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Display</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>n-1</td>
<td>n+3</td>
<td>n+4</td>
<td>n+5</td>
<td>n+6</td>
<td>n+7</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Figure 5:** Encoding, transmitting and decoding/displaying a sequence of frames with the proposed method. The sequence contains a scene cut between the frames n-1 and n.

On the next page, Figure 6 depicts the resulting improved encoding of the scene cut.
Figure 6: Sequence of 50 frames across a scene cut in one of the files used in our experiments (Claire.qcif 80-100 followed by Carphone.qcif 1-29) encoded at 32 Kbit/s with improved rate control; the bits per frame and the corresponding PSNR per frame are shown.

Even with this very simple strategy, the Table 1 shows that it is possible to achieve some gain. The Table shows the gain achieved by this simple strategy both in bits and in PSNR computed on the Y plane.

<table>
<thead>
<tr>
<th>Sequence</th>
<th>Bit Rate</th>
<th>Skipped (Total)</th>
<th>Gain (per frame on whole seq.)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>TMN-8</td>
<td>Modified</td>
</tr>
<tr>
<td>commercials</td>
<td>32 Kb/s</td>
<td>1802 (4250)</td>
<td>1831 (4250)</td>
</tr>
<tr>
<td>commercials</td>
<td>64 Kb/s</td>
<td>650 (4250)</td>
<td>631 (4250)</td>
</tr>
<tr>
<td>std</td>
<td>32 Kb/s</td>
<td>211 (4000)</td>
<td>210 (4000)</td>
</tr>
<tr>
<td>std</td>
<td>64 Kb/s</td>
<td>35 (4000)</td>
<td>34 (4000)</td>
</tr>
<tr>
<td>std100</td>
<td>32 Kb/s</td>
<td>407 (4000)</td>
<td>405 (4000)</td>
</tr>
<tr>
<td>std100</td>
<td>64 Kb/s</td>
<td>109 (4000)</td>
<td>110 (4000)</td>
</tr>
</tbody>
</table>

Table 1: Experimental results. Gain in average bit and PSNR per frame measured for the whole sequence.
Because the rate control strategy is unable to achieve the target bit rate with a sufficient precision, we have measured the results as a ratio of the encoded frame the encoded size (in bits) normalized by the PSNR achieved on the Y plane. Since the purpose of our approach is to improve the quality in proximity of scene cuts, in addition to reporting the total number of skipped frames, the gain (using the ration measure) is reported by partitioning the data into two portions:

*cuts:* The 15 frames (1/2 second) after each scene cut.

*scenes:* All other frames that are not part of the cuts.

<table>
<thead>
<tr>
<th>Sequence</th>
<th>Bit Rate</th>
<th>Skipped (Total)</th>
<th>Gain in BIT/PSNR (per frame)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>TMN-8</td>
<td>Modified</td>
</tr>
<tr>
<td>commercials</td>
<td>32 Kb/s</td>
<td>1802 (4250)</td>
<td>1831 (4250)</td>
</tr>
<tr>
<td>commercials</td>
<td>64 Kb/s</td>
<td>650 (4250)</td>
<td>631 (4250)</td>
</tr>
<tr>
<td>std</td>
<td>32 Kb/s</td>
<td>211 (4000)</td>
<td>210 (4000)</td>
</tr>
<tr>
<td>std</td>
<td>64 Kb/s</td>
<td>35 (4000)</td>
<td>34 (4000)</td>
</tr>
<tr>
<td>std100</td>
<td>32 Kb/s</td>
<td>407 (4000)</td>
<td>405 (4000)</td>
</tr>
<tr>
<td>std100</td>
<td>64 Kb/s</td>
<td>109 (4000)</td>
<td>110 (4000)</td>
</tr>
</tbody>
</table>

Table 2: Experimental results. Gain in average BIT/PSNR per frame measured for the cuts (15 frames after each scene cut), for the scenes (excluding cuts) and for the whole sequence.

As can be seen from Table 2, the effect is to improve cuts and leave scenes essentially unchanged. The atypical behavior of the sequence *commercials.gcf* when encoded at 32Kb/s occurs because our implementation uses the bit rate information from a TMN-8 encoder and applies the strategy to every sequence of skipped frames of length >1, even when it is not necessary (because there is still same space in the buffer). More sophisticated control of the publicly available H.263+ encoder we used can avoid this behavior.

### 6. Current Research

We are working on the implementation of both greedy strategies and variations of the dynamic programming optimization, that can be implemented by modifying publicly available encoders. Further theoretical research includes the detailed analysis of the consequences of eliminating the Assumption 1 from the optimal dynamic programming algorithm. It is also interesting to consider a model where frames can be predicted from ones that are arbitrarily far back in the stream (at distances greater than \(d\)); although our dynamic programming algorithm can accommodate this model, complexity and minimization of frame buffering needs to be studied.
Acknowledgment: We thank M. Cohn and F. Rizzo for many helpful discussions.

Bibliography


G. Côté, M. Gallant and F. Kossentini [1998], "Description and Results for Rate-Distortion Based Quantization", Doc. ITU-T/SG16/Q15-D-51, File Name: q15d51.doc.


