MPEG-1/VCD Digital Video Editing

Ran Bar-Sella[1], Nir Shachar, Zelig Weiner and Nimrod Peleg[2].
Signal & Image Processing Lab
Department of Electrical Engineering
Technion IIT, Haifa 32000, Israel
http:/www-sipl.technion.ac.il

Introduction

MPEG-1, and its derivative for the Video-CD market - VCD, are standards for compression of video images and their related sound track for rates of 1~1.5 Mbps. As the most wide spread standards for digital video, they both address well issues of compressing, communicating, storing and displaying the videolaudio, but lack adequate support for editing the video sequence once compressed. For bitstreams of fixed bit-rate, like VCD, the problem is even worse. Simple concatenation of two VCD bitstreams will produce an illegal VCD stream, mainly due to VBV (Video Buffer Verifier) mismatch. Editors are required to perform video editing tasks on decoded streams and re-encode the result later on. This in turn produces a loss of video quality similar to the 'loss of generation' incurred in analog video.

Our aim was to analyze the problem of performing a cut-to-cut editing on a fixed bit-rate VCD video clip without decoding and then re-encoding. The resulting video clip must be a legal VCD bitstream. The editinig process should preserve video quality and be as short as possible.

The MPEG- 1 standard separately defines the requirements for the video and audio layers as well as for the system layers which combine them both. Therefore, the editing problems were analyzed and solved for every layer separately. The proposed solutions were later combined into a unified one. This is also the structure of this paper. We first present the various problems of video editing at each layer. We then go on and present our solution for these problems. Finally we describe the way we combine all the solutions into one. For brevity we will rely heavily on the specific terms of the standard [1] [2] [3] [4] and refrain from explaining them. The reader is encouraged to use the references for glossary of terms.

Video layer editing problems

Performing a cut-to-cut editing of two VCD clips was analyzed through the following model. We take one VCD clip and select two cut points, thus cutting out a portion of the clip, resulting in two VCD clips -1 and 2. We want to join these two clips in a cut-to-cut manner. The editing process presents three problems at the video layer:

(i) Selecting a legal cut point.
(ii) Preserving syntax legality.
(iii) Preventing VBV mismatch.

The MPEG-1 standard [1] defines three types of frames - *IP & B.I* frames can be decoded independently of any other frame in the bitstream, while *P & B* frames depend on past and/or future frames. This implies a bitstream order of frames, which is different from the display order, so that the decoder will have all the frames needed to decode a specific frame even if they are to be

[1]e-mail: ranbs(ij)tx.technion.ac.il
[2]e-mail: nimrod@ee.technion.ac.il

displayed after that specific frame. This is illustrated in [1,fig.1]. The frame inter-dependence, together with the standard's requirement that the bitstrearn will not include frames which are not displayed, implies that we can not define a cut point at every frame. Selecting a cut point after a B frame in clip 1, might result in a situation where the *I*, or *P*, frame on which it depends will be part of the bitstream, so as to support the de-coding of the ***B*** frame, but will not be displayed itself. This situation is illegal under the standard. It is thus necessary to define which cut points are legal.

A simple concatenation of two VCD clips can result in a stream which does not conform with the standard's syntax due to several possible reasons. The two clips might not have been coded using the same coding parameters, e.g. source image size, aspect ratio, display rate etc. The Time Code and Temporal Reference fields of frames in clip 2 might not be valid any more due to a possible change in display timing. Additionally, the two clips might not be synchronized in their GOP status (open/close). All these are matters of syntax which must be dealt with for the resulting clip to be legal.

The main problem arising at the Video layer is a VBV mismatch. The VBV parameter is the most important tool by which MPEG- 1 controls both buffer level and encoder/coder synchronization. The encoder produces frames of *different* sizes at *fixed* frame rate and the resulting bitstream is transferred to the decoder at a *fixed* rate. This requires the use of a buffer at both the encoder and the decoder. Buffer control is implemented by having the encoder simulate the input buffer of the decoder and adjust the frame size, so over/underflow situations won't occur. Buffer level information is coded at the beginning of each frame and transmitted, as part of the bitstream, to the decoder. This is the VBV parameter and the decoder uses it to retain synchronization with the encoder. The behavior of the VBV parameter can be easily calculated using:

$$VBV_{n+1}=VBV_n-I_n+(t_{n+1}-t_n)BR \qquad (1)$$

where: In is the size of frame *n* in bits, BR is the bit-rate and $(t_{n+1}-t_n)BR$ is the number of bits which are fed into the buffer between time *n* and *n+1*. When encoding a VCD clip the encoder makes sure, through its control over I_n, that the VBV buffer will not over/underflow. When we join two VCD clips at frame *n* there is no reason to assume that either $VBV_{1,n}$ or $I_{1,n}$ of clip 1 are the same as $VBV_{2,n}$ or $I_{2,n}$ of clip 2. This implies that we can not be certain that (1) still holds around the cut point. As a result of that we can no longer be sure that the rest of the clip will not over/underflow. To visualize that, we can view the VBV behavior graphically as presented in [1, fig. C. 1]. Each time we display a frame the graph drops, at almost zero time, by the amount of bits required to encode the frame. Between frames the graph rises at a fixed slope related to the input bit-rate. When we join two clips we actually join two such graphs. At the point of concatenation we remove from the buffer the first frame of clip 2 instead of a frame of clip 1. This implies that any difference between the VBV levels of clips 1 and 2, at the cut point, indicates a problem. The original VBV level of clip 2, at the cut-point, was:

$$VBV_{2,n+1}=VBV_{2,n}-I_{2,n}+(t_{n+1}-t_n)BR \qquad (2)$$

Applying (1) for clip 2 at the concatenation point will yield the *new* VBV level of clip 2:

$$\overline{VBV}_{2,n+1}=VBV_{1,n}-I_{2,n}+(t_{n+1}-t_n)BR \qquad (3)$$

Comparing (3) to (2) we note that if $VBV_{1,n} < VBV_{2,n}$ the entire VBV graph, of clip 2, will drop lower then its original level

and thus could underflow. For similar reasons if $VBV_{1,n} > VBV_{2,n}$ the VBV graph of clip 2 , will rise and an overflow may occur. The over/underflow may occur immediately, at the cut point, or later on. Analyzing the frame sizes of VCD movies and checking a considerable amount of possible cut points for VBV difference, we concluded that in over 90% of the cases we have a possible underflow situation. Most of the time we have a VBV difference of O- 10Kbytes at the cut point. A common solution is to go over the entire second clip and check and update the VBV parameter. If an over/underflow is encountered we will also have to decode and re-encode the rest of the clip. This is a time consuming operation that sometimes involves a loss of quality due to decoding and re-encoding.

Audio layer editing problems

Editing audio layerl/2 puts no restrictions on the editing process beside the requirement to perform the audio editing at a frame aligned position. Since each audio frame can be decoded separately no problem is anticipated when joining the two streams.

System layer editing problems

At the system layer video and audio streams are multiplexed into one buffer controlled stream while retaining video/audio synchronization. The information of each stream is divided into equally sized Packs and the video/audio Packs are multiplexed, together with timing and system buffer control information, in order to retain buffer control and timing requirements [4]. Cut-to-cut editing introduces three problems at the system layer:

(i) System layer syntax legality.
(ii) STD buffer overflow.
(iii) Video/audio synchronization.

As is the case in the Video layer, when joining two clips we must address possible differences in parameters. The main issues are a possible mismatch in the display timing parameters such as PTS, DTS, and SCR. The first problem arises from the fact that while the difference between PTS's in clip 2 remains legal their absolute values are no longer so. This happens since the PTS of the cut point of clip 2 no longer has the same time reference it had when the clip was encoded. This in turn could violate the standard's requirement that the difference between two successive PTS's will not exceed 0.7 Sec.

The system layer buffer, STD - System Target Decoder, controls both the audio and video buffers through the multiplexing of audio and video Packs. The STD buffer control relies heavily on the number of Packs of each type in the past. Unlike the VBV control, the STD information is not coded in the bitstream. When we simply join two system streams we can not assume they both have the same past STD behavior and thus the STD of clip 2, used with initial condition achieved at the end of clip 1, can be expected to cause STD over/underflow. This may cause audio and/or video buffer over/underflow.

The last, but not least, of system layer problems is the loss of video/audio synchronization. The length of an audio frame -26 mSec., is different then the length of a video frame ~ 1/30 or 1/25 Sec. Since we cut both streams at frame boundaries the cut point at each stream will usually not occur at the same time reference. Joining the two clips will usually cause the audio stream of clip 2 to be shifled in time, thus losing the synchronization with the video stream.

Solutions for the Video layer

The first two problems encountered at the video layer can be easily solved by defining the following rules for cut point legality and

syntax information update (we use the display order of frames):

Clip 1: Legal cut points are after a *P* or *I* frames. Add a Sequence End Code following the cut point.

Clip 2: Legal cut points are before an *I* frame or the first ***B*** frame in a Closed GOP. Preface the clip with last Sequence Header of clip 2, followed by a Closed GOP Header. If the cut point is before an *I* frame, erase the ***B*** frames preceding it. Note that these ***B*** frames *follow* the *I* frame in bitstream order. If the first GOP has changed, update its Temporal Reference as well.

The VBV mismatch problem is solved using the *Level Adjustment Principle - LAP*, developed as part of our work. The main idea behind LAP is the observation that if we could somehow adjust the VBV level of clip 1, so that it would match the VBV level of clip 2 at the cut point we could guarantee that there will be no over/underflow problems. This is due to the fact that clip 2 is legal by itself and its initial VBV level will not be disturbed. We can further expand this principle if we note that as long as we do the VBV level adjustment along a limited number of images, and make sure we do not over/underflow along the few frames we adjust, we can still be sure clip 2 will not over/underflow. As a result; we can solve the VBV problem by simply developing tools for raising or lowering the VBV buffer contents. Lowering the buffer contents can be easily achieved by zero padding any frame of clip 1, which is legal under the standard. Raising of buffer contents, which prevents underflow, is a bit more complicated but alas a more frequent requirement. We propose several methods for raising buffer content. The first is to introduce a new frame just after the last frame of clip 1. We add a small ***P*** frame which will cause the last frame of the clip to freeze on the display for one frame cycle. Adding the small frame raises the buffer contents since we add another frame cycle. The added frame cycle time increases the buffer contents by receiving bits from the input while removing only a small amount of bits out of the buffer for the added image. Inserting a 20 byte ***P*** frame will result in a gain of 4- 10Kbyte in buffer contents. The fact that we freeze a frame which is close to the cut point prevents the viewers from noticing the freeze. We checked viewers response to frame freeze and concluded that we can introduce as much as two frames, at *each* side of the cut point, without a perceived change in the movie. Another method we propose it to reduce the frame size in bits by zeroing out several of the DCT coefficients. This is equivalent to widening the dead zone of the quantizer which reduces the image size through longer runs in the run-length coding. We developed a frame content dependent algorithm, which performs this reduction with minimal blockiness. We do so by analyzing parameters at the MB level and degrading only MB's that have limited influence on the image quality. Using these methods we can solve the VBV problem locally around the cut point, without scanning or re-encoding clip 2.

Solutions for the System layer

The fact that the STD information is not coded in the bitstream was circumvented by developing an accurate estimator for the STD level. Despite that, the strict syntax of the system layer, and especially the fixed size of Packs in VCD, introduces a serious problem while trying to meet the various syntax requirements at the system layer. We have to solve a level matching problem were our level differences can only be estimated and the level changes are limited by the

Pack size. This and the fact that we need to maintain PTS rate suggests that we can not solve the STD and syntax legality problems, at the system layer, locally. We propose to edit the video and audio streams separately and then re-multiplex them back into one system stream. Although this re-multiplexing should be performed on the entire resulting movie it is both simple and fast and does not incurs any loss of quality.

For the videolaudio synchronization problem we suggest two solutions. Noting that the cut point is selected according to the video timing we can pick the audio cut point either after or before the corresponding audio frame. The first solution is then to pick the audio cut points according to the video/audio skew introduced by the specific selection. By carefully choosing the audio cut point we can achieve a skew which is always smaller then 13 mSec. If we allow ourselves to introduce extra audio frames into the audio stream or freeze video frames we can gain more degrees of freedom in trying to re-match the sound track and the video images. We decided to limit ourselves to freezing up to two video frames ardor adding up to three audio frames. Under these restriction we can achieve a perfect synchronization in some of the cases and at the worst case a 6 mSec of video/audio mismatch. Added audio frames where taken from the original clip. Both the silence audio frames as well as the 6 mSec mismatch were found to be unnoticeable by viewers. The frame insertion method does not disturb the audio buffer control and integrates with the solutions for the VBV problem.

Combined Algorithm

We developed several methods for coping with the problems encountered while performing a cut-to-cut editing of a VCD stream. All these methods were combined into a complete algorithm for video editing.

Assume we have two system VCD clips we want to concatenate and that both cut points are legal under the cut point legality rules we have presented. In the following algorithm we put the minimization of the video/audio skew as the top priority after maintaining VCD syntax legality:

1) Demultiplex both clips into audio/video streams.
2) Perform video syntax legality adjustments.
3) Determine VBV buffer difference.
4) Determine video/audio synchronization skew.
5) Achieve minimal synchronization skew by audio cut point selection and video/audio frames insertion.
6) Adjust the VBV buffer level.
7) Re-Multiplex the streams while adjusting video temporal references fields.

Summary

We have analyzed the problems encountered while performing cut-to-cut editing of a VCD stream, at each layer of the MPEG-l/VCD standard. Solutions for all these problems were developed and integrated into one combined algorithm for performing the editing process. We retain both video and audio perceived quality as well as conform to the standard's syntax. The proposed algorithm was implemented, in C, and tested on a PC platform. A variety of input clips and cut-points were tested, producing good results.

Acknowledgment

This work was supported by Optibase Ltd.

References

1.ISO/IEC-l1172-2, MPEG-1 - video layer.
2.ISO/IEC-l1172-3, MPEG-1 - audio layer.
3.ISO/IEC-l1172-1, MPEG-1 - system layer.
4. VCD specification.