VIDEO AND AUDIO EDITING FOR MOBILE APPLICATIONS

Ari Hourunranta, Asad Islam, Fehmi Chebil

Nokia
ari.hourunranta@nokia.com, asad.islam@gmail.com, fehmi.chebil@yahoo.com

ABSTRACT

Video content creation and consumption have been increasingly available for the masses with the emergence of handheld devices capable of shooting, downloading, and playing videos. Video editing is a natural and necessary operation that is most commonly employed by users for finalizing and organizing their video content. With the constraints in processing power and memory, conventional spatial domain video editing is not a solution for mobile applications. In this paper, we present a complete video editing system for efficiently editing video content on mobile phones using compressed domain editing algorithms. A critical factor from usability point of view is the processing speed of the editing application. We show that with the proposed compressed domain editing system, typical video editing operations can be performed much faster than real-time on today's S60 phones.

1. INTRODUCTION

Almost immediately after embedded cameras were introduced into mobile phones, video-capturing applications started to emerge. Nowadays, many mobile phones can capture long clips with a reasonably good quality. This has turned mobile phones into digital camcorders. Further, using the connectivity capabilities in their devices, users can share the recorded content with friends and family instantaneously. After shooting video clips, however, users tend to trim or personalize them by introducing a set of effects, organizing them into a new sequence, removing unwanted parts or combining them with other clips.

There are several PC-based commercial products available providing such video editing functionalities. It is not very practical solution, however, to transfer files from a phone to PC for editing and then transfer the edited videos back to the phone for sharing. Editing capabilities on the mobile phones, therefore, would provide significant advantage.

Solutions from the PC world cannot be ported directly into mobile devices that are constrained by low resources in processing power, RAM memory, storage space, and battery. For these devices, decoding a video sequence and re-encoding it, typically multiple times to achieve a desired visual effect, would take significantly long time. To overcome this problem, it is necessary to utilize fast and efficient compressed domain techniques.

Efficient algorithms for video editing have been studied in the literature, mainly in the context of MPEG-1/2 videos. We show that an efficient mobile video editing system can be developed using a combination of existing and novel techniques for compressed domain editing. The existing algorithms include DCT domain editing algorithms proposed by Chang et al. [1] and temporal dependency manipulation algorithms proposed by Wee et al [2,3,4]. Further, Meng and Chang [5] presented a compressed domain video editing and parsing system with a set of editing effects, such as cutting, pasting, blending, and temporal effects.

Our complete video editing system for mobile devices offers an extensive set of features for editing video clips on mobile terminals. The application enables users to create edited movies out of their captured video content for instant sharing and playback, as well as for storing the movies in finalized shape.

The paper is organized as follows. In section 2, we look into the editing requirements in mobile environment. In section 3, we give an overview of the techniques for editing video sequences. We present the architecture of our video editor in section 4, and provide some experimental results on the performance in section 5.

2. EDITING ON MOBILE TERMINALS

In this section, we identify mobile video editing use cases and their impact on the editing procedure. We start by introducing the mobile video and audio formats employed for mobile devices and the characteristics they have for editing.

2.1. Mobile Audio and Video Formats

In the mobile domain, the most common video coding formats for user-generated content today are ITU-T H.263 baseline [6] and ISO MPEG-4 Simple Profile [7]. They both are based on the traditional DCT and motion compensation based hybrid coding scheme with intra (I) and inter (P) pictures.

For the associated audio, the most relevant formats are 3GPP's AMR-NB and MPEG-4's AAC. The relevant characteristic for editing in both of them is that the frames are independently encoded, which enables cutting and splicing audio streams at individual frame boundaries.

The file formats used in mobile use cases – 3GP and MP4 – are inherited from ISO media file format [8], making them closely compatible. The ISO media file format is based on a concept of separated metadata and media data. The metadata contains common information for the whole file, and detailed information of video and audio frames, e.g. timestamps, grouped as video and audio tracks. The frames can be located and read based on the information in the audio/video tracks, as illustrated in Figure 1.

The original use case for mobile-generated video was Multimedia Messaging Service (MMS). MMS puts rather strict restrictions for video clips (e.g. codecs, bitrates, video resolution, clip size), but the idea is to enable MMS usage even in low-end phone categories. The 3GP multimedia file format was designed to include H.263 video and AMR-NB audio for MMS use.

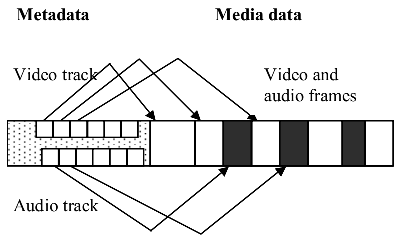

Figure 1: ISO/3GP/MP4 file format structure

2.2. Use Cases

2.2.1. Video Sharing

The video editing application should be able to generate videos that can be shared over the MMS. To meet the MMS restrictions, and in general to save up/downloading time, video clips may require size reduction – a process that typically requires user interactions. In addition to reducing video resolution and increasing compression, the clip may need to be shortened, and the user must be provided a way to control how to cut the clip.

To optimize the use of the limited size, the editor should provide a possibility to create a summary clip, by picking up only the relevant parts of the clips and splicing them.

2.2.2. Video Content Creation

Video recording capabilities of today's high-end smart-phones are getting close to the camcorders. In real camcorder use cases, it is justified to use MPEG-4 Simple Profile video and MPEG-4 AAC audio instead of the 3GPP formats.

For camcorder-type of video content, video editing could support users in creating useful video clips that are worth storing, such as, documents of holiday trips or birthday parties. This can mean removing unwanted scenes, splicing video clips, and adding titles or some effects. Further, inserting an image (still image, text frame) inside a video can enrich the video.

A natural requirement for a video application is to be able to use video content independently of its origin. This is not straightforward, since as explained above, mobile videos can be created for various formats, depending on the original use case.

The above mentioned use cases and requirements imply that a number of editing operations are needed for an efficient video editing solution on mobile devices. The video editing tool must essentially be able to support video cutting and splicing, as well as video transcoding features. It must support transitional effects (such as Wipe, Fade, etc.) as well as some special effects (such as Slow Motion, Black & White, etc.). Moreover, it must provide basic editing support for the audio associated with video clips.

3. EFFICIENT ALGORITHMS FOR EDITING OPERATIONS ON MOBILE TERMINALS

In this section, we give an overview of the video editing operations that our system supports; details are presented in [9].

3.1. Video Editing Operations

The following editing operations were selected based on the identified use cases, and were optimized for mobile devices – splicing, cutting, transitional effects for fading, slow motion, black & white, inserting still images in videos, and basic audio effects.

Splicing and cutting can be considered as the very basic editing operations. Both operations modify durations of video, and

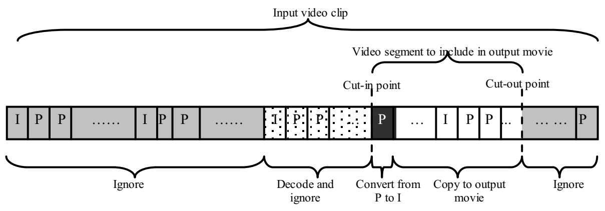

Figure 2: Cutting procedure of a video clip

therefore it is essential to ensure the timing information remains continuous over the editing points and consistent for all frames.

Splicing can be done in compressed domain by simply concatenating the video bitstreams, translating timestamps, and updating the file format metadata. However, splicing two videos in compressed domain is possible only if the format and resolution of the two videos are the same. If they are not, one clip must be transcoded to the format and/or resolution of the other clip.

In cutting a video, the principle proposed by Wang and Woods [10] for MPEG-2 streams can be applied. Video frames are first decoded starting from the last preceding I-frame up to the first frame to include in the output, which is then converted to an I-frame. The timing information of all the included frames is then converted to map the translation in time due to cutting These steps are illustrated in Figure 2. The frequency of I-frames in the input has direct impact to the speed of cutting operation.

For performing transitional effects, such as fading in and fading out, we use a hybrid approach, where only the transition frames are re-encoded while the rest of the video is processed in compressed domain. The transition part can be considered as a new input clip, first cut out from the video, then transcoded with the effect applied on spatial domain, and then spliced with the rest of the video.

Still images can be inserted into video by decoding the input JPEG image and then encoding it as a video clip, either as a single I-frame with a given duration, or as a set of video frames. The resulting video clip is then spliced with the other video clips.

In addition to the basic trimming type of operations, special effects can also be applied to video. Slow Motion effect can be achieved by changing the timing information of the clip. Theoretically, the same approach could be used for fast motion effect too, but care must be taken not to increase the playback frame rate above the limits set by the standards. Black & White Effect can essentially be achieved by simply removing chrominance data from the compressed video bitstream.

An important thing to note is that in contrast to the traditional transcoding-based editing that repetitively employs lossy coding, compressed domain editing retains original quality of video while simultaneously providing significant processing speed-ups.

3.2. Audio Editing Operations

Different kinds of audio operations can be employed to support the video editing system. Our system supports the following three simplest audio editing operations: retaining, replacing, and muting.

Retaining simply copies the audio from input video clips to output movie. The cut points of the video and audio must match exactly in order to avoid any audio drift in the edited video clip.

Replacing is used to add new audio, e.g. a music file. It may require transcoding the audio to a compatible format.

Audio can be muted by inserting "silent" audio frames that give the effect of silence.

4. EDITING ARCHITECTURE

In this section, we present a complete editing system that can be built based on the proposed editing operations. Our video editing system employs both system components and internal editing components, as illustrated in Figure 3.

The input and output to the system are files while the user controls the operations. The heart of the editing system consists of two key modules: Video Processor and Audio Processor. All the components of the video editing system are discussed below.

4.1. System-Level Components

The system-level components are not specific to the editing system but can be standard components. For example, if the editor is implemented on S60 phones, hardware accelerated codecs can be used through Symbian APIs, namely Multimedia Framework (MMF) and Media Device Framework (MDF).

4.1.1 File Format Parser and Composer

File format parser is used to extract metadata information (such as video/audio duration, frame properties), and to read compressed video and audio frames from input 3GP or MP4 files. Similarly, the composer creates output 3GP or MP4 files using the generated metadata information and the edited video and audio frames.

4.1.2 Video and Still Image Codecs

Video decoder is used to decode compressed video to spatial domain, whereas video encoder encodes spatial domain video to compressed signal. While this editing system does most of the processing in compressed domain, there are instances when full decoding of a video frame is needed, for example when seeking for a cutting point or when applying a transition effect. Similarly, the video encoder is needed only in some cases, for example when applying transition effect, inserting still images, or converting P frame to I in cutting. Both of them are needed in full transcoding as well. In the still image insertion case, still image decoder is used instead of video decoder.

In cases where data is modified in compressed domain, the editor must also be capable of doing partial decoding and encoding operations, such as VLC coding, since typically hardware accelerators do not provide access to such individual operations.

4.1.3. Audio Codecs

Audio encoder and decoder are used in our system only when transcoding audio.

4.2 Video Processor

Video Processor provides the core of the editing system. It consists of several components, as illustrated in Figure 4 and listed below, and it interfaces with the introduced system level components.

4.2.1. Frame Analyzer

Frame Analyzer takes in the information about the video frame, and in conjunction with the editing parameters, decides the kind of operations to be performed on the frame. It may remove the frame altogether, or it may feed the frame to the decoder for full decoding. Alternatively, it may send the frame to the compressed domain processor for editing the frame in compressed domain.

4.2.2. Spatial Domain Processor

The Spatial Domain Processor is used to perform some spatial domain processing on the raw video frame, e.g. for the transitional effect, or for scaling in transcoding case.

4.2.3. Compressed Domain Processor

Compressed Domain Processor performs compressed domain editing operations on the video frame, based on the information provided by the Frame Analyzer. The operations include, for example, Black and White effect and time stamp alignment.

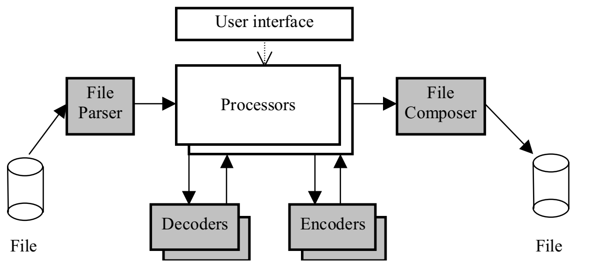

Figure 3: High-level architecture of the video editing system

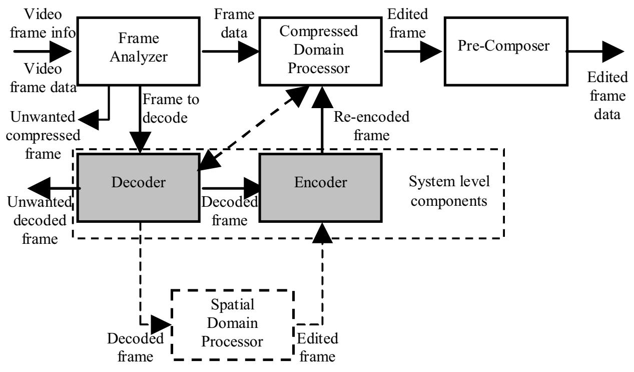

Figure 4: Video Processor module

4.2.4. Pre-Composer

Once the compressed video frame data is ready after editing, Pre-Composer collects and updates the file format information. This information consists of video frame size and type, timestamp, etc.

4.3. Audio Processor

Audio Processor provides simple audio editing features to support the video editing system. It processes the audio data in the input clips in accordance with the editing parameters to generate the desired audio track in the output movie, keeping the timing information synchronized with the video. The processed audio frames are sent to the Composer for inclusion in the output movie.

5. EXPERIMENTAL RESULTS AND DISCUSSION

To verify the efficiency of the presented compressed-domain editing solutions, we compared the processing times of the editing operations to the durations of the video clips, i.e. how much faster than real-time can an edited video clip be generated. The rationale is that a spatial domain editing system on mobile device can typically only encode video in real-time and, hence, cannot generate edited video in less time than the video duration.

We measured the performance of the developed video editing system on an OMAP1710-based S60 imaging phone. The phone is capable to separately record and playback MPEG-4 CIF video in real-time, but it cannot do CIF transcoding in real-time. The operations involving video encoding utilized a HW-accelerated video encoder; all other processing was done in software on ARM processor. The operating speed is naturally very dependent on the hardware, optimization level of the software, and how well the system layer parameters are tuned for editing type of operations. However, running the tests on a commercial product should at least give an idea of the achievable performance. However, similarly to the PC world, the HW performance of imaging phones is continuously improving, and the results present only a snapshot of the situation at the time of conducting this study.

Table 1 shows the test cases and the ratios of the editing speeds and clip durations. Ratios greater than 1.0 indicate the operation is faster than real-time. For example, ratio 10 means that a 60-second video can be edited in 6 seconds. We selected four cases that should be typical for mobile video editing: a simple cutting case, a simple splicing case with cutting and a transitional effect, a splicing case with special effects and a transitional effect, and a complicated case with title frame generation and insertion, video cutting, still image insertion, and several transitional effects. All the input video clips were about 60 seconds long. None of the test cases involved transcoding of video to another resolution or format, since that is mainly dependent on the performance of the system layer components. However, all involved partial video decoding and partial video encoding, especially the last case where more than 25% of the clip was encoded, and the 2.7 Megapixel still image was decoded and scaled to video resolution. Input clips had I-frames about every five seconds. The clips also contained audio track, which was retained in the output.

Table 1 Test cases and the relative editing speeds compared to real-time operation

Test cases	**H.263 QCIF 15 fps @ 128 kbps**	**MPEG-4 CIF 15 fps @ 512 kbps**
Video (Cut 24…54s)	14.9 x	4.4 x
Video1 (Cut 12…42s) + Fade + Video2 (Cut 12…42s)	7.8 x	2.1 x
Video1 (Black & White) + Wipe + Video2 (Slow Motion)	6.4 x	1.7 x
Title (5 s) + Fading + Video (Cut 7…52s) + Crossfading + Still image (10 s) + Fading	2.5 x	1.0 x

The results show that typical editing operations for video clips recorded on mobile phones can be performed at least in real-time on the phone, and in most cases, much faster. Especially in the MMS-compatible QCIF-resolution, editing is significantly faster than real-time. The reported figures can, however, still be improved by software optimizations even on the same HW.

In cases where compressed domain algorithms are dominating the editing process, editing MPEG-4 CIF clips is about 4 times more complex than editing H.263 QCIF clips. The performance is mainly dependent on bit-rate, since the operations are done on bitstream layer. Therefore, operations like bitstream parsing and shifting, VLC coding, and file I/O operations may become the bottlenecks of the system, whereas in traditional video processing the bottlenecks are elsewhere, like in IDCT/DCT transform and in pixel processing related to motion estimation and compensation. Further, typical codec accelerators do not provide access to individual operations. Therefore, utilizing hardware acceleration to further improve the performance of compressed domain video editing may not be straightforward, but requires at least additional APIs to access individual operations on hardware.

6. CONCLUSION

In this paper, we addressed the problem of providing video editing capabilities on memory- and power-constrained portable devices, such as mobile phones with integrated video cameras. In order to avoid the computationally demanding decode-edit-encode cycle for a typical video editing session, we proposed a viable alternative for video editing on mobile phones — by developing a highly optimized, compressed domain video editing system. This system supports most of the common video editing features, such as video cutting and splicing, frame insertion, special effects (such as slow motion and black & white), transitional effects (such as wipe and fade), and supporting audio editing capabilities. Results indicate substantial gains in editing times for these operations, without loss of quality, making it viable to perform video editing on today's mobile phones.

As the mobile technology expands its focus more towards the multimedia market, it is imperative that efficient and viable solutions are provided for mobile devices for managing the growing multimedia content. Mobile video editor is one such step in this direction.

7. REFERENCES

[1] S.F. Chang and D.G. Messerschmitt, "Manipulation And Compositing Of MC--DCT Compressed Video", *IEEE Journal on Selected Areas in Communications: Special Issue on Intelligent Signal Processing*, 13:1--11, 1995.
[2] S. J. Wee and B. Vasudev, "Splicing MPEG Video Streams In The Compressed Domain", *IEEE Workshop on Multimedia Signal Processing*, Princeton, NJ, Jun 1997.
[3] S. J Wee and J. Apostolopoulos, "Efficient Processing Of Compressed Video", *Asilomar Conference on Signals, Systems, and Computers*, Volume 1 , 1-4, pp. 853 – 857, Nov. 1998.
[4] S. J Wee, "Manipulating Temporal Dependencies In Compressed Video Data With Applications To Compressed-Domain Processing Of MPEG Video", *IEEE International Conference on Acoustics, Speech, and Signal Processing*, Phoenix, AZ, Vol. 6, pp. 3129-3132, March 1999.
[5] J. Meng and S.-F. Chang, "CVEPS: A Compressed Video Editing and Parsing System", *ACM Multimedia Conference*, Boston, MA, Nov. 1996.
[6] ITU-T Recommendation H.263, "Video Coding For Low Bit Rate Communication", February 1998.
[7] ISO/IEC JTC 1/SC 29/WG 11 N4350, "Information Technology – Coding Of Audio-Visual Objects – Part 2: Visual", July 2001.
[8] ISO/IEC JTC 1/SC 29/WG 11 N4350, "Information Technology – Coding Of Audio-Visual Objects – Part 12: ISO Base Media File Format", February 2004.
[9] A. Islam, F. Chebil, and A. Hourunranta, "Efficient Algorithms for Editing Videos on Mobile Terminals", submitted to *ICIP 2006*, Atlanta, GA, Oct. 2006.
[10] K. Wang and J.W. Woods, "Compressed Domain MPEG-2 Video Editing", *IEEE International Conference on Multimedia and Expo 2000*, Volume 1, 30 July-2 Aug. 2000, pp. 225-228.