J. Vis. Commun. Image R. 25 (2014) 689–697

Contents lists available at SciVerse ScienceDirect

J. Vis. Commun. Image R.

journal homepage: www.elsevier.com/locate/jvci

# Systematic analysis of the decoding delay in multiview video

Pablo Carballeira *, Julián Cabrera, Fernando Jaureguizar, Narciso García

*Grupo de Tratamiento de Imágenes, ETSI de Telecomunicación, Universidad Politécnica de Madrid, 28040 Madrid, Spain*

ARTICLE INFO

*Article history:*
Available online 19 April 2013

*Keywords:*
Three-dimensional video
Multiview video coding
Video conference
Low latency
Decoding delay
Parallel processing
Graph theory
Process scheduling

ABSTRACT

We present a framework for the analysis of the decoding delay in multiview video coding (MVC). We show that in real-time applications, an accurate estimation of the decoding delay is essential to achieve a minimum communication latency. As opposed to single-view codecs, the complexity of the multiview prediction structure and the parallel decoding of several views requires a systematic analysis of this decoding delay, which we solve using graph theory and a model of the decoder hardware architecture. Our framework assumes a decoder implementation in general purpose multi-core processors with multi-threading capabilities. For this hardware model, we show that frame processing times depend on the computational load of the decoder and we provide an iterative algorithm to compute jointly frame processing times and decoding delay. Finally, we show that decoding delay analysis can be applied to design decoders with the objective of minimizing the communication latency of the MVC system.



## 1. Introduction

For several years, video technologies have targeted the development of systems that provide immersive viewing experiences. Nowadays, the advances in three-dimensional (3D) display technologies have made 3D video an emerging and sustainable market in the near future. 3D Video (3DV) and free viewpoint video (FVV) are new types of visual media that expand the user's experience beyond what is offered by 2D video [1], providing a 3D depth impression of the scene, and interactive viewpoint selection. Currently, these types of visual media systems are beginning to enter into consumer markets, such as entertainment and mobile applications [2]. For those systems, a data format that is richer than single 2D video signal is needed. The spectrum of data formats for 3D Video goes from purely image-based data formats like multiview video (multiple views of the same scene) to data formats related to computer graphics like 3D meshes and their corresponding textures [3]. A widely adopted approach is the one that includes multiview video and depth sequences as additional scene geometry information, allowing the possibility of generating additional views on virtual camera positions [4]. Nevertheless, the size of this multiview video grows linearly with the number of views while the available bandwidth is generally limited. Thus, an efficient compression scheme for multiview video is needed.

Multiview video coding (MVC) [5] is an extension of the H.264/MPEG-4 Advanced Video Coding (AVC) standard [6] that provides efficient coding of such multiview video. Besides, as depth signals can be represented as monochromatic video signals, MVC has been also commonly used to compress them [4]. As an extension of AVC, MVC makes use of the set of AVC coding tools. The key additional feature of the MVC design, that increases the coding efficiency specifically for multiview video, is a new prediction relationship between frames of different views that exploits the interview redundancy. This prediction relationship is known as interview prediction. Fig. 1 shows a sample prediction structure in which temporal and interview predictions are used.

MVC allows a wide range of applications and scenarios [7]. Here, we address real-time applications such as live broadcasting, videoconferencing or interactive streaming [8] where constraints on the end-to-end delay are imposed. The one-way delay between both ends of the conversation is known as *communication latency*, i.e., the delay between the instant when a frame is captured and the instant when it is displayed at the receiver. In bidirectional applications, the constraint on communication latency is stricter. For those, typical recommendations on maximum communication latency generally state that there is none or little impact below 150 ms, while a serious impact may be observed above 400 ms [9].

Each element (encoder, transmission channel and decoder) contributes to the delay between the instant when a frame is captured and the instant when it is decoded at the receiver: the *system delay*. For each frame, the value of the system delay varies due to different factors, such as the required encoding time or the nature of the transmission channel (variable or constant bitrate, packet losses, etc.). Since frames have to be displayed at a constant rate, generally receivers utilize an output buffer for decoded frames, to guarantee a constant communication latency. In practice, this buffer results in an additional variable delay for each decoded frame: the *display delay*. Therefore, the communication latency is the sum of the system delay and the display delay. In real-time applications, the design of

* Corresponding author.
*E-mail address:* pcl@gti.ssr.upm.es (P. Carballeira).

1047-3203/$ - see front matter © 2013 Elsevier Inc. All rights reserved.
http://dx.doi.org/10.1016/j.jvcir.2013.04.004

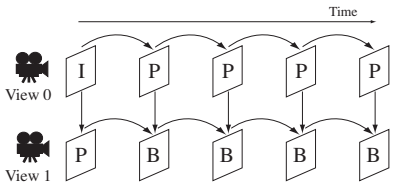


**Fig. 1.** Example of a multiview prediction structure for two cameras. Horizontal arrows correspond to temporal prediction and vertical arrows to interview prediction.

this output buffer, and the display delay is a challenging issue. On the one hand, the display delay should be as minimum as possible since it increases the communication latency. On the other hand, it has to be high enough to absorb the variability of the system delay so that frames are displayed at a constant rate. While in non-live services, such as video on demand, this display delay may be over-dimensioned with little impact on the service, this requirement is stricter in the case of real-time bidirectional services. Thus, an accurate computation of the system delay is essential to design a system with a minimum valid display delay.

In [10], we presented a framework for the analysis of the encoding delay for MVC. Now, in this paper, we focus on the analysis of the contribution of MVC decoders to the system delay: the *decoding delay*. Our purpose here is to provide tools for an accurate evaluation of the decoding delay in order to complete the analysis of the contribution of the MVC codec processes to the system delay. The decoding delay in MVC decoders depends on two different but related factors:

1. The multiview prediction structure: temporal and interview prediction relationships among frames establish decoding order dependencies for a frame.
2. The hardware architecture and implementation of the decoder: specific architectural features of multiview decoders (e.g. number of processors, use of threads etc.) influence the time needed to decode a given frame, and therefore, they affect the decoding delay performance.

Whereas in single view decoders, the computation of the decoding delay can be easily approximated as the decoding time of one frame, in the case of MVC, the complexity of multiview prediction structures, and the presence of several views that need to be decoded simultaneously, increase the complexity of the decoding delay analysis. Thus, we present here a framework for the systematic analysis of the decoding delay in MVC decoders. This framework evaluates the decoding delay taking into account: (i) the multiview prediction structure and (ii) the hardware implementation of the decoder. Nowadays, actual decoders support several parallel streams and different codecs, and the general tendency is to incorporate general purpose processors, in which the decoders are software-implemented, instead of traditional dedicated hardware processors. This tendency is particularly interesting to handle MVC streams due to its inherent parallelization characteristics [11,12]. Therefore, our framework assumes a hardware platform for the decoder based on a multi-core processor with multi-threading capabilities. We define a *decoding process* as the set of operations that are needed to decode a frame. Our model assumes that any decoding process can run on an exclusively dedicated core (processor from now on) or one of the threads that share the processing power of one of the processors. The required time to run that process will be higher if several processes share the same processor.

Analogously to the encoding latency analysis [10], we rely on graph theory to compute the decoding delay for this hardware model. A graph is constructed from the multiview prediction structure in which the frames can be seen as the nodes and the prediction dependencies as the edges. Each edge has an associated cost that represents the contribution of the prediction dependency to the decoding delay. We show that frame processing times depend on the computational load of the decoder and we provide an iterative algorithm to compute jointly frame processing times and the decoding delay by an iterative analysis of the graph.

In our results, we use the decoding delay analysis to characterize the communication latency of a complete MVC system. We show that this analysis can be used to determine hardware requirements of MVC decoders, such as number of processors or processor throughput (number of frames that one processor is able to decode per second), with the objective of achieving a target communication latency. For example, we show that for a given processor throughput, the decoding delay can be reduced by increasing the number of processors in the decoder, until certain limit that we can identify. Increasing the number of processors above that limit does not further decrease the decoding delay.

This paper is organized as follows: in Section 2, we discuss the communication latency of an MVC system and the role of the decoding delay on it. In Section 3, we present our framework for the decoding delay analysis in a multi-thread decoder architecture. In Section 4 we present the iterative algorithm for the computation of processing times and decoding delay. In Section 5 we show the experimental results and in Section 6 we present the conclusions.

## 2. Discussion on communication latency of MVC systems

As aforementioned, the communication latency indicates the time elapsed between the instant when a frame is captured, and the instant when that frame is displayed. A block diagram of an MVC system and the elements that add to the communication delay between its both ends, are depicted in Fig. 2. For frame $x_j^i$ (frame $j$ of view $i$), $t_{capt_j^i}$ is the instant when $x_j^i$ is captured, $t_{cod_j^i}$ is the time instant when $x_j^i$ is completely coded, $t_{RX_j^i}$ is the instant when the

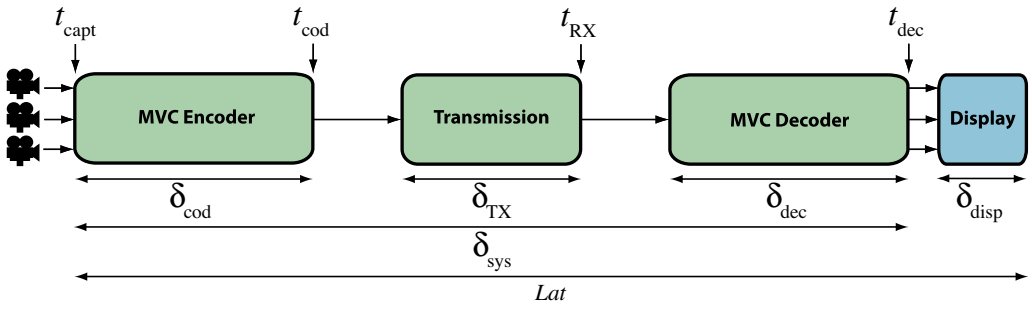


**Fig. 2.** Block diagram of the MVC system with encoding delay, transmission delay, decoding delay, system delay, display delay and communication latency.