AN EFFICIENT VIDEO RENDERING SYSTEM FOR REAL-TIME ADAPTIVE PLAYOUT BASED ON PHYSICAL MOTION FIELD ESTIMATION [1]

Luca Piccarreta, Augusto Sarti, Stefano Tubaro

Dipartimento di Elettronica e Informazione, Politecnico di Milano
Piazza Leonardo da Vinci 32, 20133 Milano,ITALY
phone: + 39-02-2399-3647, fax: +39-02-2399-3413
email: luca.piccarreta, augusto.sarti, stefano.tubaro@elet.polimi.it
web: www-dsp.elet.polimi.it

ABSTRACT

In this paper we present an implementation of an Adaptive Playout System for video rendering. This system can be used for rendering multimedia material that is delivered (in single/multi-cast fashion) to the final user(s) over a "best-effort" network that is unable to guarantee a constant delay in the delivery of the data packets. The proposed solution is an alternative to the "traditional" pre-buffering at the decoder side, which "cushions" the variations of delivery delay, but forces the final user to wait a great deal before the rendering starts and may easily generate annoying freezing of the video in case of pre-buffer underflow. The system can be quite effective in enabling the "zapping" between channels broadcasted over the network by using streaming technology. The system is currently able to run in real time on commercial PCs for the decoding and adaptive playout of CIF sequences, but there is still a great deal of room for further software optimization. A series of informal subjective tests have been conducted to demonstrate the potential of the system.

1. INTRODUCTION

Thanks to the formidable rate of growth of the Internet, an increasing number of network applications are being used today by the average users. Among the real-time applications, IP telephony, voice conferencing, Internet radio, and Video on Demand (VoD), have become widely used. The Internet, however, does not allow us to easily handle real-time traffic, as the packet transmission quality (e.g., transmission delay, jitter, and packet loss) may vary quite dramatically. In order to compensate for variable delays in real-time applications, it is customary to use a smoothing buffer at the client side. After several packets are piled up in the buffer, the actual decoding may start (*playout delay*). This way the influence of the delay variations within the network can be minimized. The choice of the playout delay is important because it directly affects the communication quality of the application. If the playout delay is set too short, the client application often treats packets as lost even if they eventually arrive. On the other hand, a large playout delay may easily become unacceptable to the final users. The problems related to playout control have been considered quite recently [1], but the Adaptive Media Playout (see below) for media streaming systems is still relatively unexplored.

It is important to notice that the retransmission of lost media packets can be essential for video streaming over error-prone channels. Due to the interdependency of successive video packets introduced by motion compensated prediction in modern video encoding schemes like MPEG-2, MPEG-4, or H.264, continuous and reliable audio and video playout at the receiver can only be guaranteed if all the packets are available at the receiver side.

The media streaming systems strive to allow the immediate fruition of media data as it is delivered from a remote server. In practice, however, the systems must buffer an amount of media at the client to prevent packet losses and delays. System designers must trade the reliability of uninterrupted playout against delay when determining the amount of data to buffer. Designers of today's commercial media streaming products find, for example, that buffering delays ranging from 5 to 15 s provide a good balance between delay and playout reliability [2]. In contrast, viewers accustomed to traditional broadcast television expect playout to be immediate and program changes to be instantaneous.

Adaptive media playout (AMP) allows the client to buffer less data and, thus, introduces less delay to achieve a given playout reliability. In this scheme, the client varies the rate at which it plays out audio and video according to the state of its playout buffer. Generally, when the buffer occupancy is below a desired level, the client plays media slowly to reduce its data consumption rate. Slowed playout will cause viewing latency to increase. In this case, faster-than-normal playout can be used during good channel periods to eliminate any excess latency accumulated with slowed playout. By manipulating playout speeds AMP can reduce initial buffering delays in the case of prestored streams, and reduce the viewing latency of live streams, all without sacrificing playout reliability. To control the playout speed of media, the client scales the duration of each video frame as shown, and processes audio [3] to scale it in time without affecting its pitch. Variations in the media playout rate are found to be subjectively less irritating than playout interruptions and long delays [4]. Informal tests have shown that playout speed variations of up to 25% are often not noticeable and,

[1] The work here presented has been carried out in the framework of the EU funded IST-Project: FP6-507452 (MEDIANET)

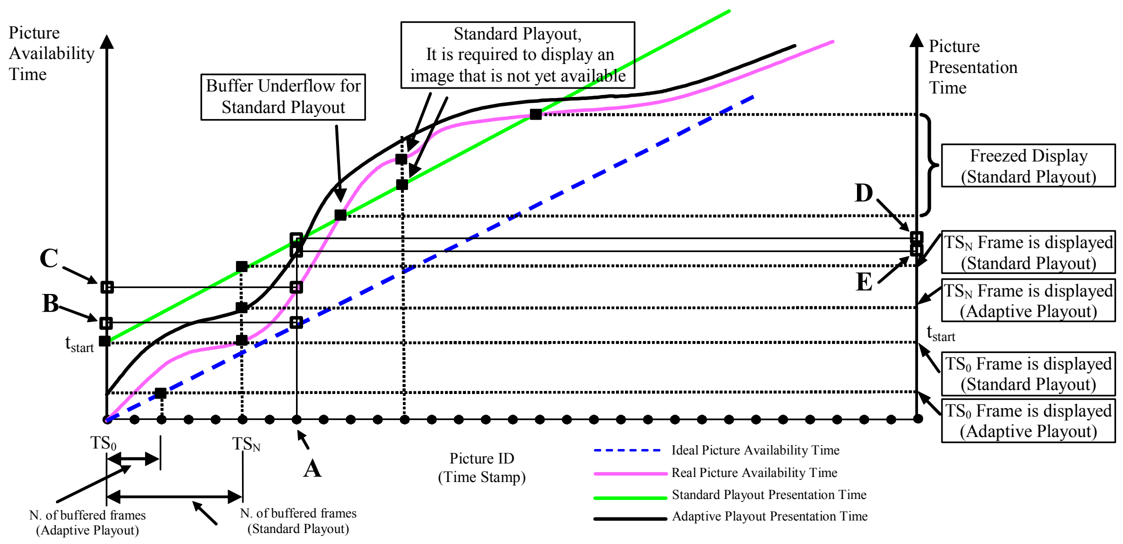


Figure 1. Adaptive Playout vs Standard Playout (with variable data delivery delay). How to read the figure. A: Frame that we consider; B: Ideal time at which the frame is available at the decoder output; C: Effective time (due to variable network delay) at which the frame is available; D: Standard Playout, time at which the frame is displayed; E: Adaptive Playout, time at which the frame is displayed.

depending on the content, rate variations up to 50% are sometimes acceptable [4]. Moreover, it is important to notice that media playout speed modification is widely used in TV productions.

The combination of adaptive speech playout and time-scale modification has recently been proposed for packet voice communication [3], while in [5] adaptive playout of video as well as audio data are used for relaxing the stringent delay constraints imposed for real-time voice transmission so that multiple retransmissions of lost packets can be afforded.

Several approaches have been used for the analysis of the AMP systems and for defining their performances. For example, in [6] and [4] different models are used for characterizing and simulating the transmission network in order to judge (with subjective tests) the performance of the AMP systems. The results are always positive for the use of AMP systems instead of simple playout with interruption (in case of lost data) or with significant delays when very large buffers are used.

2. DESCRIPTION OF THE PROPOSED SOLUTION

In a complete AMP system both the video and the audio components of a TV signal must be treated in a synchronized fashion in order to compensate for the variable network delay and the data packet loss. The processing of the video, however, is indeed the most computationally demanding task, therefore we focused our attention on the adaptive playout of video content only. In traditional playout schemes, media rendering begins only after an appropriate number of coded pictures has been buffered (see Fig. 1). Let t_{start} (on the time axis t) be the time when the first picture (with time-stamp TS_o) is available to the decoder and displayed on the terminal. This allows us to define the new time axis:

$$t_{client} = t - t_{start} + TS_0 \quad .$$

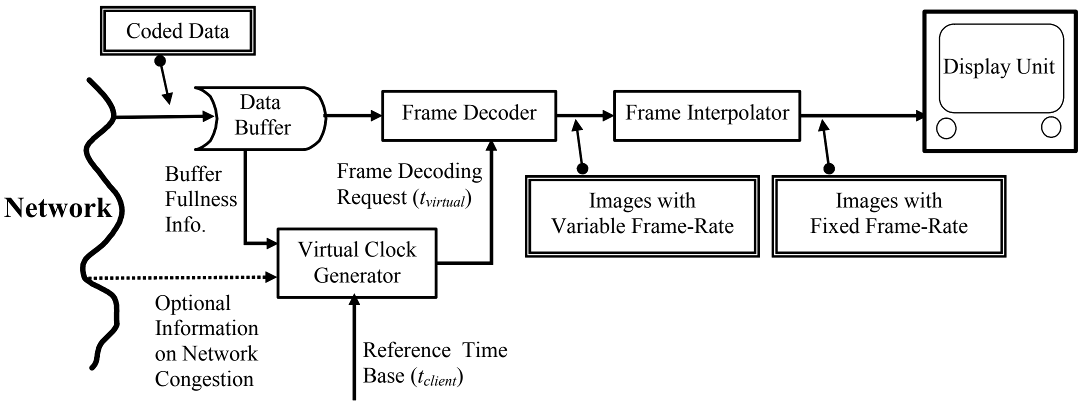
In traditional playout schemes, the frame rendered at a given time (t) is the one whose time-stamp is the highest one that does not exceed t_{client}. This approach leads to the fact that, in the presence of large network delays, the rendered video could "freeze" (see Figure 1). The adaptive playout is aimed at warping the t_{client} temporal axis in such a way to render the output video in a smoother fashion. This is done through the generation of a "virtual clock"

$$t_{virtual} = t_{client} + \Delta t_{adaptive} \quad . \qquad (1)$$

The $\Delta t_{adaptive}$ term, may vary over time driven by the decoder-buffer fullness and (optionally) by information on network congestion (see Figure 2). Using the virtual clock the presentation time of a generic picture, identified by the TS_n time stamp, can be defined as the time instant in which $t_{virtual} = TS_n$.

Using a standard playout approach the buffering performed at the client (decoder) side, before that video playout starts, can be used for compensating small fluctuations of the data packet arrival, but if the buffer is not well dimensioned, a picture that it is not yet available could be required to be displayed. In an adaptive playout scheme, $t_{virtual}$ (the time base used to identify the time stamp of the image to display) is influenced by the data buffer fullness and this guarantees that the request of displaying a frame that is not available (usually called buffer underflow) may not occur.

The display accepts images at a fixed frame rate, while the adaptive playout system provides a variable frame rate. We thus need a frame interpolator to provide the display with the correct input. Each image is generated by this block using two (or more) of the last processed frames. Figure 2 shows a complete block description of the video adaptive playout system that we propose.



Coded Data
Data Buffer
Frame Decoder
Frame Interpolator
Display Unit
Network
Buffer Fullness Info.
Frame Decoding Request ($t_{virtual}$)
Images with Variable Frame-Rate
Images with Fixed Frame-Rate
Virtual Clock Generator
Optional Information on Network Congestion
Reference Time Base (t_{client})

Figure 2: Block diagram of a Video Adaptive Playout System

MC Frame Interpolation - Motion compensated image interpolation is carried out in a variety of applications, such as video format conversion (PAL/NTSC). Whenever we need to interpolate a new image into an intermediate location between two given frames, in order to limit the artefacts due to aliasing each image element should move along the space-time trajectory determined by its motion vector. This defines its location on the interpolated image. Unlike motion estimation for image compression, motion estimation for frame interpolation should closely represent the physical motion of the objects that are present in the scene. A motion estimation that only considers the matching of luminance profiles, in fact, could easily estimate non-physical motion information, which could lead to interpolation artefacts that are perceived as even more annoying than the simple temporal repetition or averaging of the available frames.

The approach used in our motion estimation is a modification of the well-known block-based motion estimation algorithms, which tries to preserve motion physicality.

Given two images $I_t(i,j)$ and $I_{t+1}(i,j)$, in order to interpolate an image at time $t+\Delta t$ ($0<\Delta t<1$), we need to estimate the motion field at that time. Given a block on the intermediate image, we search for a motion vector **mv** that defines the two displacements: $-\mathbf{mv}\cdot\Delta t$ and $\mathbf{mv}\cdot(1-\Delta t)$. These displacements identify two corresponding blocks on the available images, which will be used for interpolating the one on the intermediate frame.

Motion Estimation is performed using an iterative optimisation process. For each block, motion vector estimation is carried out by minimizing the SAD (Sum of Absolute Difference) between the two blocks identified by the current motion vector and the similarity of the ***mv*** with those associated to the neighbouring blocks. We start from a block dimension that 1/16 (in each direction) of the original image size and then these blocks are split down to 2x2 pixels. After each block reduction, the motion field computed by using the parent block is used to initialise the new estimation. For each block dimension the minimisation process is iterated several times over all the blocks in order to best reach the minimum of the cost function. Motion estimation for large blocks is performed on downsampled versions of the input image. The resolution of the estimated motion vectors is limited, for a computational efficiency reason to a ¼ of pixel. As far as the MC interpolation of the intermediate frame is concerned, we chose to interpolate each block on the intermediate frame by using a weighted sum of the two blocks, on the known images, identified by the estimated motion vector. We call these blocks forward and backward "fathers" of the current block.

For all the regions of the image to interpolate in which the two father blocks are not in agreement (regions where the estimated motion model fails) a fall-back interpolation solution (a zero motion model) is adopted. A typical case in which this type of fall-back solutions becomes crucial is when logos are superimposed to a moving image area.

3. EXPERIMENTAL RESULTS

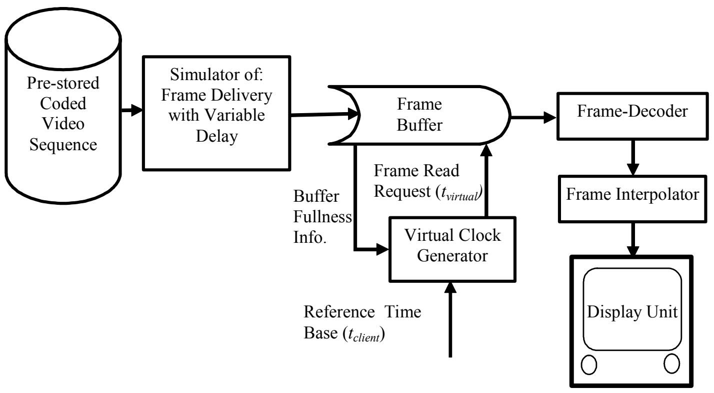
In order to test the proposed Video Adaptive Playout system we decided to use an Intel-PC platform, and we implemented a simplified version of the system, whose block diagram is shown in Fig. 3 (left). The sequence to display is stored in a file as AVC-Baseline coded material, therefore the implemented scheme includes an AVD decoder.

A block that simulates the variable delivery delay of a network is employed as the starting elements of the simulated chain. A graphical interface (Fig. 3 - right) is used for setting and changing (in real time) the network delay.

In order to control the rate used for reading the coded images from the system buffer (see Fig. 3 – left) we used a simple equation that tends to preserve the level of buffer fullness. Referring to Eq. (1), we have:

$$\alpha(t) = \frac{frames\_in\_buffer(t)}{optimal\_buffer\_fullness} - 1;$$

$$\Delta t_{adaptive}(t+1) = \Delta t_{adaptive}(t) + \begin{cases} Th & when\ \alpha(t) > Th \\ \alpha(t) & when\ |\alpha(t)| \le Th \\ -Th & when\ \alpha(t) < Th \end{cases}$$


Pre-stored Coded Video Sequence
Simulator of: Frame Delivery with Variable Delay
Frame Buffer
Frame-Decoder
Frame Read Request ($t_{virtual}$)
Buffer Fullness Info.
Virtual Clock Generator
Frame Interpolator
Display Unit
Reference Time Base (t_{client})

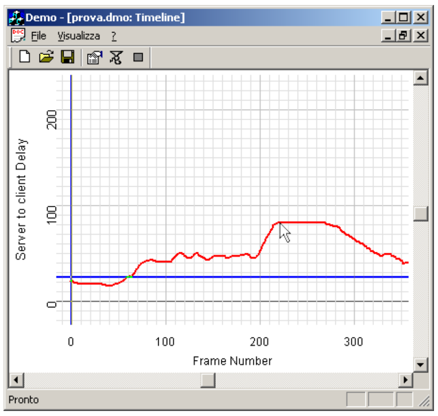


Figure 3. Left: Block Diagram of the simulated Video Adaptive Playout System. Right: User interface of the block that simulates the variable delay in delivering the video data to the client.

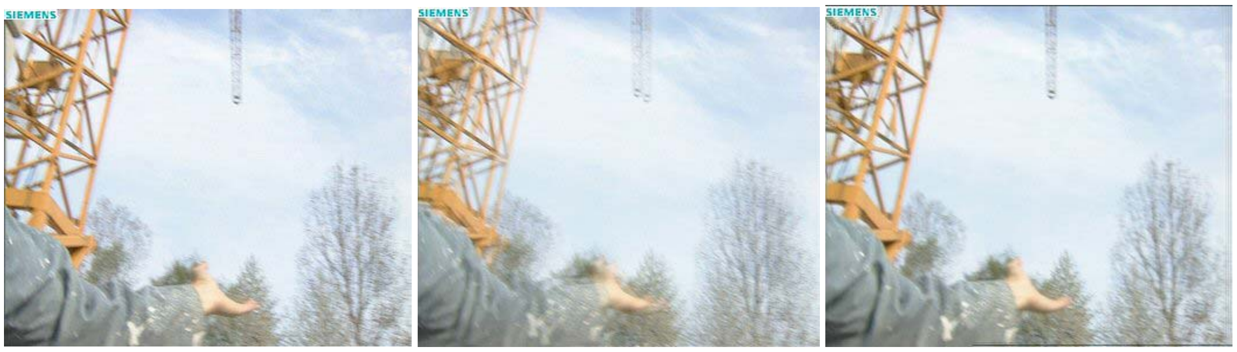


Figure 4. "Foreman" Sequence. Left: An original image, Center: An image interpolated by using a fading from the two more near available images. Right: An image interpolated by using a motion compensated strategy.

where K represents the gain of the feedback channel of the reaction and Th sets the range of the playout speed variation.
The system works in real-time with video at CIF-resolution using a modern Intel_Processor@1.5GHz ore more. In particular, in order to be able to operate in real-time, the system works (for image interpolation) on 4x4 pixel blocks with motion vectors with a ½ pixel of resolution. Some images that show the quality of the interpolated frames are shown in Figure 4.
The developed system has been used to test how the final users prefer the adaptive video playout of a streamed video with respect to a standard playout with very large buffer. A set of informal tests carried out with a number of people in our premises have shown that the adaptive video playout is largely preferred when the transmission network introduces variable and large delays in the delivery of data packets.

4. CONCLUSIONS

In this paper we presented an efficient and effective Adaptive Video Playout System that is able to run in real-time on a PC platform. Informal subjective tests have confirmed the performance improvement of the proposed approach with respect to a standard playout.

REFERENCES

[1] S. B. Moon, J. Kurose, and D. Towsley, "Packet audio playout delay adjustment: performance bounds and algorithms," ACM/Springer Multimedia Systems, vol. 5, pp. 17–28, January 1998.

[2] Windows Media Player [Online]. Available: http://www.windowsmedia.com.

[3] Y. J. Liang, N. Färber, and B. Girod, "Adaptive playout scheduling using time-scale modification in packet voice communication," in Proc. ICASSP, vol. 3, Salt Lake City, UT, May 2001, pp. 1445–1448.

[4] M. Kalman, E. Steinbach, and B. Girod, "Adaptive media playout for low-delay video streaming over error-prone channels," in IEEE Trans. on CSVT, Vol. 14, No. 6, June 2004, pp. 841-851.

[5] A. Stenger, K. Ben Younes, R. Reng, and B. Girod, "A new error concealment technique for audio transmission with packet loss," in Proc. EUSIPCO, Trieste, Italy, Sept. 1996, pp. 1965–1968.

[6] E. G. Steinbach, N. Färber, and B. Girod, "Adaptive playout for low latency video streaming," in Proc. Int. Conf. Image Processing, vol. 1, Thessaloniki, Greece, Oct. 2001, pp. 962–965.