Temporal Transcoding for Mobile Video Communication*

Maurizio A. Bonuccelli†
Dipartimento di Informatica
Via Buonarroti 2, Pisa, Italy
bonucce@di.unipi.it

Francesca Lonetti†
Dipartimento di Informatica
Via Buonarroti 2, Pisa, Italy
lonetti@di.unipi.it

Francesca Martelli
ISTI - CNR
Via Moruzzi 1, Pisa, Italy
f.martelli@isti.cnr.it

Abstract

Third generation mobile communication systems will provide more advanced types of interactive and distribution services, and video is one of the most prominent applications for multimedia communications. Adapting the media content to different networks characteristics (communication links and access terminals), in order to enable video delivery with acceptable service quality, is one of the most important problems in this setting. In this paper, we consider one of the video adaptation methods, namely video transcoding, and we present new buffer-based strategies for temporal video transcoding in a real-time context. Simulation results show that our strategies achieve a good performance in hard transcoding conditions also.

1 Introduction

Third generation mobile communication systems (e.g. UMTS) offer new and attractive services (as video streaming, video telephony, video conference) to mobile users. These services involve different types of devices and communication links. A fair and flexible allocation of the limited radio bandwidth resources among different types of services, with their respective quality requirements, is a critical issue.

A typical strategy to approach this problem is the *content adaptation*, better known as *transcoding*, usually performed by servers of a communication system, or by gateways interconnecting different networks. Transcoding allows users to encode, transmit and decode according to their features (such as channel bandwidth and terminal complexity) and preferences (such as desired video quality). Video transcoding is the process of converting a video sequence into another one with different features, without totally decoding and re-encoding, so by reducing the complexity and the running time, and enabling the interoperability of heterogeneous multimedia networks [3]. Video transcoding can provide *format conversion*, resolution scaling (*spatial transcoding*), bit rate conversion (*quality transcoding*), frame rate conversion (*temporal transcoding*). We are interested in temporal transcoding. In order to distribute the same encoded video sequence to users through channels with different bandwidths (as for instance in a multicast session), the coded video sequence must be converted into specific bit rates for each outgoing channel. This is needed also when the bandwidth of a channel is temporarily reduced for accommodating additional users when all channels are busy (subrating). Temporal transcoding does this by eliminating some frames in the sequence, in order to reduce the frame rate of the video sequence, without decreasing the video quality of not skipped frames. When frames are skipped, recomputing the motion vectors (since the old ones are no longer valid because they refer to skipped frames) and the prediction errors (for the same reason), is in order. This is done by re-using the motion vectors and the prediction errors in the input video sequence as much as possible. In addition, a frame skipping strategy must be adopted, that is a policy for deciding the frames to be dropped. We propose a new buffer-based frame skipping policy to allow real time communication. To refine this policy, other three skipping policies to be used together with this one, are proposed.

The paper is organized as follows. In Section 2, we address the temporal transcoding problem, we survey the results present in literature and we describe our temporal transcoder, able to support real time communications. In Section 3 we present our frame skipping strategies. Experimental results are drawn in Section 4. Finally, conclusions and future work are highlighted in Section 5.

2 Temporal transcoding

The typical strategy adopted in motion vectors computation is *Motion Vector Composition* (MVC) [1, 5, 6, 8], together with a restricted motion estimation called *Refined Search* (RS).

*This work has been supported by Ericsson Lab Italy, within the PisaTel Lab at ISTI - CNR.

†Also at ISTI - CNR, Via Moruzzi 1, Pisa, Italy.

About frame skipping policies, there are in literature several proposals. In [4, 5], two different strategies based on *motion activity* are presented. Motion activity gives a measure of the motion in a frame, and is defined as the sum of the motion vector components in that frame. If the motion activity is larger than a given threshold, the frame is not skipped, since it has considerable motion, and so transcoding this frame improves the smoothness of the video sequence. Another policy considering the variation of motion activity between the sequence where a frame is transcoded and the sequence where that frame is skipped, has been proposed in [7]. A different approach has been pursued in [2], where a rate control mechanism based on a buffer level prediction algorithm is proposed.

In order to meet the needs of real time applications, our main goal was to study temporal transcoding techniques guaranteeing a fixed communication delay. To this end, a transcoder output buffer is introduced. Before describing our transcoder architecture, we give some definitions . The words "input" and "output" are always related to the transcoder. We call IR the input bit rate, and R the output bit rate; ρ indicates the frame rate of the input video sequence. S and L are the size and the occupancy of the transcoder buffer, respectively. With $l(f)$, we denote the size of the transcoded frame f. Disregarding transmission time, the delay τ introduced in the communication system, is determined by L/R: in this way, the maximum delay incurred by a data bit of the transcoded video sequence is at most S/R. We choose a maximum delay of $\tau = 500ms$[1], that is considered the maximum admitted delay of a real time communication. In order to meet τ, we set the buffer size S to half the output bit rate R.

We developed a temporal transcoder architecture able to reduce the input bit rate IR of the incoming video sequence, by eliminating some frames, so that the output bit rate R turns out to be constant. Notice that the frame rate of the output video sequence is not constant, and we assumed that the skipped frames are replaced by the previous ones (*freezing*) at the displaying time in the final decoder.

In our transcoder, the motion vectors are computed by one of the four MVC algorithms[2], previously mentioned, and RS procedure. The prediction errors are computed in the pixel domain. We need to transcode each frame of the input video sequence, before applying the skipping policies, since, as we shall see in the following, they need to know some features of the reconstructed frame (for instance, its size). Transcoding each frame is needed also for avoiding to store all skipped frames between the current frame and the last transmitted one, which implies large memory resources. Reconstructed frames are then skipped or placed in the buffer for being transmitted.

[1] Our arguments are still valid also with lower values of τ.

[2] In our experiments we observed that the results obtained by these four algorithms are equivalent.

3 Frame skipping policies

The main concern of this paper is the frame skipping problem, and we present new policies which aim to meet the real time constraint, as well as to achieve a good video quality. In order to meet the first objective, a basic policy, based on buffer occupancy, is developed. The other ones, are associated to the previous, and consider other measures such as the motion activity, the number of consecutive skipped frames, and a random choice. In the following, we describe all policies in detail.

3.1 Buffer Occupancy

In order to guarantee a fixed communication delay, considering the buffer occupancy in frame skipping is needed. We present a buffer-based frame skipping policy where two buffer thresholds, B_{lower} and B_{upper}, are established for avoiding buffer underflow and overflow. Underflow occurs when the buffer occupancy is zero, and so the final decoder receives data of a frame after it is scheduled to be displayed, causing the stop of the video sequence (besides the non-utilization of the communication bandwidth). Buffer overflow occurs when the buffer occupancy exceeds the buffer size, and it increases the assumed delay τ. This is equivalent to a frame loss at the decoder, since at displaying time some bits of the corresponding frame are still in the transcoder output buffer waiting to be transmitted. B_{lower} and B_{upper} are dynamically set according to the ratio IR/R. We observed experimentally that the best values for B_{lower} and B_{upper} are respectively 20% and 80% of the buffer size when $IR/R = 2$. If $IR/R > 2$ it is needed to decrease B_{upper} so that the free buffer space is always (in average) sufficient to accommodate at least one frame. For instance, when $IR/R = 4$, a good value for B_{upper} is 60%. A frame is skipped if the buffer occupancy is greater than $B_{upper}S$, and it is always transcoded if the buffer occupancy is lower than $B_{lower}S$. Independently from the value of the threshold, in our buffer-based policy, we avoid the buffer overflow by testing that the size of the transcoded frame does not exceed the free buffer space. The only exception is for the first frame, which is an intra frame, and it is always transcoded. If the size of the first frame exceeds the buffer size, we have an additional delay equal to τ_0 for those bits which do not fit in the buffer, and after an initial delay of $\tau + \tau_0$, this frame skipping policy guarantees a constant delay τ for the whole transmission. If the output bit rate is equal to R, and a constant frame rate ρ is used, we assume that the buffer occupancy decreases at a constant rate of R/ρ bits every $1/\rho$ seconds. The whole procedure is described by the following pseudo-code.

Basic Policy(frame f):
if (f = first frame) transcode f
else
  if (($L \leq B_{lower}(S)$)&($L + l(f) \leq S$)) transcode f
  else
    if ($L \geq B_{upper}(S)$) skip f
    else
      if($L + l(f) \geq S$) skip f
      else transcode f OR **apply one of the next policies**

In the next sections, we describe three policies that can be applied at the last step of the above procedure.

3.2 Motion-based frame skipping

In Section 2, we reported some motion-based frame skipping policies proposed in literature. We present here a new motion based frame skipping policy that is applied when the buffer constraints are met. The goal of this policy is to transcode the frames with high motion. To perform this, a new motion activity (MA) measure is introduced. We slightly modified the definition given in [5], and proposed the following one:

$$MA = \sum_{m} k^{|x_m|} + k^{|y_m|} \quad (1)$$

where m is a macroblock, k is a properly tuned constant and x_m and y_m are the motion vector components of macroblock m. In this way, the motion activity measure assumes large values both in case of frame with many but small motion vectors and in case of frames with few but large motion vectors. These two cases correspond to different kind of motion: the first one occurs when there are little movements of many objects; the second occurs when there are few objects with great motion. Moreover, since an intra macroblock is produced when there are many prediction errors (namely, the macroblock is largely different from the reference area in the previous frame), we assign to intra macroblocks the maximum motion activity value, equal to the maximum size of the motion vectors, which corresponds to the search range used by the Motion Estimation procedure. In this way, we take into account of intra macroblocks also in the motion activity computation. If a frame has a small value of motion activity, it can be skipped since it is well replaced by the previous frame. Otherwise, it has considerable motion, and it should be transcoded. In our motion-based frame skipping policy, the motion activity of a frame is compared with a threshold Thr. The threshold $Thr(f)$ is dynamically set to take into account (with equal weight) the motion activity of the previous transcoded frame $MA(f-1)$, and the motion activity of all earlier frames $Thr(f-1)$. The motion-based frame skipping policy is shown in the following pseudo-code.

Motion-based Policy(frame f):
if(f = first frame) $Thr(f) = 0$;
else $Thr(f) = (Thr(f-1) + MA(f-1))/2$;
if ($MA(f) \leq Thr(f)$) skip f
else transcode f

This policy can lead to an high number of skipped frames, since it skips many consecutive frames having a low value of motion activity.

3.3 Consecutive frames skipping

This policy has been developed for attempting to overcome an harmful problem arising in *hard transcoding conditions*, that is when an high variation between the input and the output bit rate occurs (from 128 Kbit/s to 32 Kbit/s, for instance). Given that the input bit rate is much greater than the output one, it is unavoidable to consecutively skip many frames, since their size is large with respect to the output channel bandwidth. By skipping many consecutive frames, the size of the transcoded ones increases, since their motion vectors and prediction errors are obtained by adding those ones of the skipped frames. So, it can happen that the size of a transcoded frame exceeds the free buffer space. Thus, if that frame is transcoded, buffer overflow occurs, but if it is skipped, the size of the next transcoded frame will be larger. Even if, in the meanwhile, the free buffer space increases, it could not be sufficient to accommodate the transcoded frame. So, it is possible to reach an irreversible situation, in which if the frame is transcoded, buffer overflow occurs, but if it is skipped, buffer underflow occurs. We propose a solution for this problem, by trying to minimize the number of consecutive skipped frames. This is done by forcing the transcoder to drop a frame (even if its transcoding does not cause buffer overflow), in order to prevent that many frames are dropped later.

We define $\Gamma = IR/R$ representing the ratio between the input and the output bit rate. Ideally, if all transcoded frames keep their original size and have the same size, the number of transcoded frames should be equal to $1/\Gamma$. Let N be the total number of frames in the sequence. The temporal transcoder should transcode $N(1/\Gamma)$ frames and skip $N(1-1/\Gamma)$ frames. Every Γ successive frames, one of them should be transcoded, and $\Gamma - 1$ should be skipped for distributing uniformly the skipped frames. This strategy forces the transcoder to skip $\Gamma - 1$ consecutive frames, in order to prevent the number of consecutive skipped frames to become larger than $\Gamma - 1$.

We show below the pseudo-code of the whole strategy.

MaxConsecutiveSkipping Policy(frame f):
if (numConsecutiveSkippedFrames $< \Gamma$)
  skip f;
  numConsecutiveSkippedFrames++;
else
  transcode f;
  numConsecutiveSkippedFrames=0;

However, this policy does not guarantee that the above critical situation never happens, but it is very unlikely.

3.4 Random frame skipping

Randomization is used for studying the behavior of a system when input data do not follow any known law. In our setting, the sizes of incoming frames are variable and it is not possible to assume a certain distribution. This motivated us to try managing the frame skipping in a randomized way. As we saw in Section 3.1, in real time setting, the temporal transcoder choices firstly depend on the buffer occupancy. We design a simple random strategy based on the buffer occupancy, in order to decide what frames are to be skipped. We uniformly generate a random number in the range $[0..S]$. If this number is larger than the buffer occupancy L, the current frame is transcoded, otherwise it is skipped. We observe that the greater is the buffer occupancy, the smaller is the probability that the random number is larger than occupancy, so the smaller is the probability of transcoding the frame. In this way, we try to transcode more frames when the free buffer level is high, and to skip more frames when the buffer occupancy is high. We show below the pseudo-code of this strategy.

Random Policy(frame f):
randomNumber = random() % S;
if (randomNumber $\geq L$) transcode f
else skip f.

In the next section, we show the results of our frame skipping policies.

4 Simulation results

We implemented an MPEG4-based transcoder and evaluated the performance of our frame skipping strategies by considering two metrics: the number of transcoded frames (indicating the video sequence smoothness), and the PSNR (a measure that indicates the quality of transcoded sequence by taking into account the differences of the luminance values of corresponding pixels in the original and reconstructed frame). We compute the PSNR between the transcoded video sequence and the video sequence decoded after the front encoder. Two kinds of PSNR measures are considered: the first one, that we call PSNR1, takes into account of transcoded and skipped frames, by replacing these last with their previous ones (freezing). In the second, that we call PSNR2, only transcoded frames are considered. Given that our transcoder is a purely temporal (and not a quality) one, quality degradation is due to frame dropping only. So, the first way to compute PSNR allows us to measure the actual visual quality perceived by the final user. The second way indicates the quality of single transcoded frames, without capturing the degradation introduced by frame dropping.

	mobile			foreman			coastguard		
	Frames	PSNR1	PSNR2	Frames	PSNR1	PSNR2	Frames	PSNR1	PSNR2
Standard Transcoding condition									
Buffer	155	27.09	29.21	144	30.08	34.01	105	28.72	34.36
MA-based	145	25.73	28.34	127	28.08	33.73	106	27.66	33.70
Consecutive	149	26.58	28.52	134	29.81	33.97	96	28.47	34.01
Random	148	25.95	28.72	132	28.43	33.13	106	28.13	34.13
Hard Transcoding condition									
Buffer	59	22.84	28.02	45	24.21	35.00	35	24.06	35.32
MA-based	60	21.38	27.77	50	23.57	33.71	32	23.95	34.25
Consecutive	57	22.80	28.02	47	24.21	33.92	34	23.95	33.97
Random	59	22.52	27.95	50	24.36	33.84	34	24.11	34.26

Table 1. Number of transcoded frames, PSNR (dB), PSNR2 (dB), for different video sequences in standard/hard transcoding conditions.

We consider several video sequences in QCIF format and frame rate of 30 fps. We show only the most significant experimental results about different benchmark video sequences of 300 frames: "mobile", which is a video sequence with a lot of motion, "foreman", which is a video sequence with scene changes, and "coastguard" where there are moving objects. We evaluated our frame skipping strategies both for "standard" and "hard" transcoding conditions[3]. For the first case, we consider $IR = 128$ kbps and $R = 64$ kbps; for the second one, $IR = 128$ kbps and $R = 32$ kbps. We report in Table 1 the average PSNR1 and PSNR2 as explained above, and Figures 1 and 2 show the PSNR1 of the first 50 frames for "mobile" sequence.

In order to have a real-time communication, buffer occupancy is the dominant factor, that is why it is considered in all the frame skipping strategies. Consequently, from our experimental results we deduce that there are not large differences on the PSNR achieved by different frame skipping strategies (see Figures 1 and 2).

By looking at the top of Table 1 we observe that all strategies reduce to about one half the number of frames, so achieving the same ratio between R and IR for "mobile" sequence, while for other sequences the number of transcoded frames is lower. In the bottom of Table 1 we report the results for hard transcoding: we note that "consecutive" skipping policy behaves similarly to the "buffer-based" policy, in terms of average PSNR, but by looking at Figure 2, we observe that, in hard transcoding conditions, the "consecutive" policy is better than the others, since the PSNR is smoother. This happens because the frames are dropped more uniformly.

5 Conclusions

We implemented four frame skipping strategies in order to improve the quality of temporal video transcoding

[3] With "standard" transcoding conditions we mean a typical situation in which the transcoder output channel has a bandwidth equal to an half of the input bandwidth.

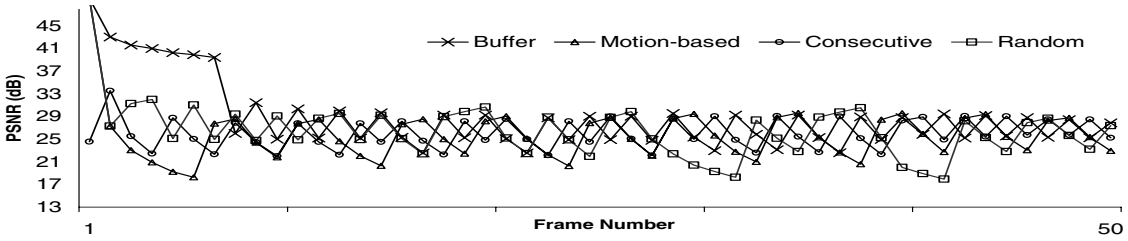

Figure 1. PSNR of our frame skipping policies for "mobile" video sequence ($IR = 128$, $R = 64$ kbps).

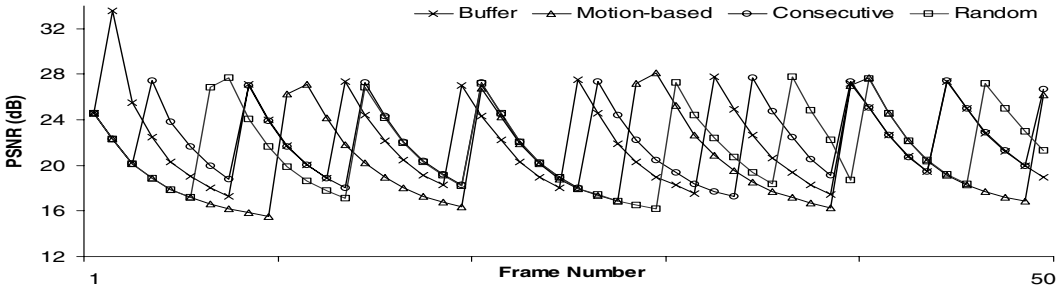

Figure 2. PSNR of our frame skipping policies for "mobile" video sequence ($IR = 128$, $R = 32$ kbps).

in a real-time environment. Disregarding the transmission time, we obtained a real time communication with a fixed admitted delay. In the "buffer occupancy" strategy, we achieved this by considering only the buffer occupancy to skip frames, and avoiding buffer underflow and overflow. In the others, we considered other metrics in order to improve the visual quality. There are not large differences on the PSNR achieved by the proposed frame skipping strategies, but the "consecutive" policy achieves a better visual quality in hard transcoding conditions, since skipping an high number of consecutive frames is avoided.

Several problems are still open. An interesting one is an analytical study of the buffer occupancy over time, in order to reduce the maximum admitted delay τ. Besides, we intend to refine the parameters of "motion-based" and "random" strategies, by means of an extensive simulation phase. Another interesting issue is to test the behavior of our policies on H.263-based transcoder, and on the emerging H.264-based one.

Acknowledgment

We thank Ericsson Lab Italy team working on video transcoding, in particular Giovanni Iacovoni and Salvatore Morsa for introducing us in this research area, and for helpful discussions.

References

[1] M.-J. Chen, M.-C. Chu, and C.-W. Pan. Efficient motion-estimation algorithm for reduced frame-rate video transcoder. *IEEE Trans. on Circuits and Systems for Video Technology*, 12(4):269–275, Apr. 2002.

[2] P.D.F. Correia, V. Silva, and P.A. Assunção. A method for improving the quality of mobile video under hard transcoding conditions. In *Proc. of IEEE ICC03*, Vol. 2, pp. 928–932, Anchorage, Alaska, USA, May 2003.

[3] S. Dogan, A.H. Sadka, and A.M. Kondoz. Efficient MPEG-4/H.263 video transcoder for interoperability of heterogeneous multimedia networks. *Elettronics Letters*, 35(11):863–864, May 1999.

[4] K.-T. Fung, Y.-L. Chan, and W.-C. Siu. New architecture for dynamic frame-skipping transcoder. *IEEE Trans. on Image Processing*, 11(8):886–900, Aug. 2002.

[5] J.-N. Hwang, T.-D. Wu, and C.-W. Lin. Dynamic frame-skipping in video transcoding. In *Proc. of IEEE* 2^{nd} *Workshop on Multimedia Signal Processing*, pp. 616–621, Redondo Beach, CA, USA, Dec. 1998.

[6] T. Shanableh and M. Ghanbari. Heterogeneous video transcoding to lower spatio-temporal resolutions and different encoding formats. *IEEE Trans. on Multimedia*, 2(2):101–109, Jun. 2000.

[7] H. Shu and L.-P. Chau. Frame-skipping transcoding with motion change consideration. In *Proc. of IEEE ISCAS 2004*, Vol. 3, pp. 773–776, Vancouver, Canada, May 2004.

[8] J. Youn, M.-T. Sun, and C.-W. Lin. Motion vector refinement for high-performance transcoding. *IEEE Trans. on Multimedia*, 1(1):30–40, Mar. 1999.