Temporal Transcoding for Mobile Video Communication*

Maurizio A. Bonuccelli[(1,2)] Francesca Lonetti[(1,2)] Francesca Martelli[(2)†]

(1) Dipartimento di Informatica, Via Buonarroti, 2, 56127 Pisa, Italy
Ph: +39.050.2212755/3108 Fax:+39.050.2212726
(2) ISTI-CNR, Via Moruzzi, 1, 56124 Pisa, Italy
Ph: +39.050.3153468 Fax: +39.050.3152924
e-mail: {bonucce,lonetti}@di.unipi.it, 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.

Video delivery in heterogeneous communication environments is enabled by one of the following three strategies [3]. *Simulcast distribution* is the delivery of independently encoded copies of the same video content, each one compliant to varying features, such as bit/frame rates and spatial resolution. The disadvantage of this strategy is the high number of copies of the same video, which implies transmission bandwidth and storage resources over usage. *Scalable media model* provides a base layer for minimum requirements, and one or more enhancement layers, to offer improved qualities at increasing bit/frame rates and resolutions. This strategy leads to overall video quality degradation with the increasing level of scalability, particularly when the base layer is encoded at a low bit rate. Furthermore, it requires layered encoding and decoding capabilities in sender's and receiver's devices, difficult to implement in low-power mobile terminals.

The third strategy is *content adaptation*, better known as *transcoding*, typically 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 [4]. Video transcoding could provide *format conversion*, resolution scaling (*spatial transcoding*), bit rate conversion (*quality transcoding*), frame rate conversion (*temporal transcoding*). Format conversion operates a syntax change from a video coding standard to another one, for instance, from MPEG encoded video stream to H.263 encoded one.

Spatial transcoding, i.e. reduction of the spatial resolution of the compressed video is required for facing the problem of limited size in many access terminals. The simplest case is the 2:1 downscaling [8, 9]. Recent works investigate the spatial transcoding with arbitrary down-sampling ratio [10, 12].

In order to distribute the same encoded video sequence to users through channels with different capabilities, the coded video sequence must be converted into specific bit rates for each outgoing chan-

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

†Contact author.

nel. Quality transcoding does this by operating on the bit allocation for each frame and by tuning the quantization parameters of every macroblock of the frame according to the target bit rate. The consequence of this is a variable frame quality. When the bandwidth in a wireless network is very limited, the quality transcoding process can cause high degradation of the transcoded video quality if the frame rate is constant.

Temporal transcoding is a process that eliminates 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.

In a real time setting, the buffer control is a critical issue, and it is influenced by the number of consecutive dropped frames. We propose a new buffer-based frame skipping policy allowing real time communication. We propose also other three skipping policies to be used together with the buffer-based one: the first is based on the amount of motion present in the frames, with the goal of dropping frames with less movements; another which attempts to limit the number of consecutive skipped frames, in order to avoid (as we shall see in the following) an irreversible situation in which a purely temporal transcoder is not able to produce output. The third one, is a random strategy, that has the advantage of being simple, but statistically good.

We implemented our temporal transcoder with MPEG4 codec, and we evaluated the performance of our frame skipping strategies by considering two metrics: the number of transcoded frames (indicating the smoothness of the video sequence), and the PSNR (indicating the quality of transcoded frames). We observed that constant output bit rate, real time constraints, and good quality are provided by all strategies.

The paper is organized as follows. In Section 2, we address the temporal transcoding problem and survey the results present in literature; in Section 3, we describe our temporal transcoder, able to support real time communications. In Section 4 we present our frame skipping strategies. Experimental results are drawn in Section 5. Finally, conclusions and future work are highlighted in Section 6.

2 Temporal transcoding

As said before, in temporal transcoding three issues must be addressed: the computation of the motion vectors, the computation of the prediction errors, and the frame skipping policy.

For the first problem, the typical strategy is the *Motion Vector Composition* (MVC), together with a restricted motion estimation called *Refined Search* (RS). For each macroblock, a candidate motion vector is computed by composing the motion vectors of all dropped frames between the current frame and the last not dropped frame. The new motion vectors are then obtained by searching around the candidate motion vector obtained by the MVC, within a few pixels search area. In literature, four MVC algorithms are known: Bilinear Interpolation [7], Telescopic Vector Composition [9], Forward Dominant Vector Selection [13], and Activity Dominant Vector Selection [1].

There are in literature two ways for computing the new prediction errors: a standard one (the same used by the encoder), is by computing, in the pixel domain, the differences between the current macroblock and the reference area, in the last not skipped frame, pointed by the *new* motion vector (obtained by MVC and RS). Then, these differences are encoded with the usual DCT and quantization. An alternative way [5, 6], is to add the errors of the current macroblock to those of the macroblock or reference area in the previous skipped frame. According to our experiments, the first strategy guarantees better video quality but it requires a greater computation time. The second one is faster, since it does not perform DCT, quantization and inverse DCT and inverse quantization, when the motion vector of the current frame points to a macroblock in the previous skipped frame (this typically happens with null motion vectors). So its prediction errors are already present in the input coded bit stream. When the motion vector points to a reference area which does not overlap a macroblock in the previous skipped frame, the prediction errors of this area are not available in the input data. The re-quantization introduced for computing these errors brings additional re-enconding errors. Such errors degrade the quality of the reconstructed frame. Since each non skipped frame is used as reference for the following non skipped frames, quality degradation propagates to later frames in a cumulative manner. When more frames are dropped, this degradation cannot be entirely avoided, even if error-compensation strategies are applied.

About frame skipping policies, there are in literature several results. A strategy based on *motion activity* has been presented [7]. The motion ac-

tivity gives a measure of the motion in a frame. The motion activity of a frame t, $MA(t)$, is defined as the sum of the motion activities $MA(t)_m$ of all macroblocks of the frame t, where $MA(t)_m$ is the sum of the horizontal and vertical components of the motion vector of macroblock m:

$$MA(t)_m = |x_i| + |y_i|. \quad (1)$$

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 strategy [5] has been developed for facing the problem of the re-encoding errors, when the prediction errors are computed in the alternative way described before. The goal of this strategy is to minimize the re-encoding errors as well as to preserve the motion smoothness. It is based on a metric which is the motion activity of the current frame divided by the sum of the re-encoding errors. If this is greater than a given threshold, the frame should not be skipped because it has considerable motion. On the contrary, if this metric is smaller than the threshold, the frame can be skipped since it contains many re-encoding errors. The threshold can be dynamically set according to the target frame rate of the transcoder.

A control scheme considering the motion change and trying to reduce the jerky effect caused by frame skipping has been proposed [11]. At each frame, the motion change is given by the difference, in terms of motion vectors, between the sequence where that frame is transcoded and the sequence where that frame is skipped, and replaced by the previous one during the decoding phase. An high value of motion change of a frame causes an evident jerky effect if that frame is skipped. In addition, frames are skipped also according to buffer occupancy, in order to achieve a constant output bit rate. Another policy is proposed in [2]; it consists in a rate control mechanism based on a buffer level prediction algorithm, in order to reduce the number of consecutive skipped frames which generates the jerky effect.

3 Our temporal transcoder

In current communication systems, many advanced multimedia applications have real time features. In order to meet the needs of such real time applications, our main goal was to study temporal transcoding techniques guaranteeing a fixed communication delay. In order to perform this, 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 the transmission time, the delay τ introduced into 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 S/R. We choose a maximum delay of $\tau = 500ms$, 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, in such a way 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 addition, our temporal transcoder guarantees real time, by limiting to τ the maximum delay incurred by data.

In our transcoder, the motion vectors are computed by one of the four MVC algorithms[1], described in Section 2, and RS procedure. The prediction errors are computed in the standard way described above. We need to reconstruct the motion vectors and the prediction errors for 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). Reconstructed frames are then skipped or placed in the buffer for being transmitted. Notice that the transcoding of each frame is needed in case of both transmission and dropping, since the successive frame has to be transcoded in terms of the previous one in both cases. This is needed also for avoiding to store all skipped frames between the current frame and the last transmitted one, which implies large memory resources.

4 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 achieve

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

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.

4.1 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 in order to avoid buffer underflow and overflow. Underflow occurs when the buffer occupancy is zero and so the final decoder receives data of the video 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 set dynamically 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 first frame, that is an intra frame, 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 another policy*

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

4.2 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 measure is introduced. We slightly modified the definition given in equation 1 in the following way:

$$MA_m = k^{|x_i|} + k^{|y_i|} \quad (2)$$

where k is a properly tuned constant. 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 a lot of 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

As we shall see, this policy can lead to an high number of skipped frames, since it skips many consecutive frames having a low value of motion activity.

4.3 Consecutive frames skipping

When the bandwidth of a coded video sequence needs to be drastically reduced, namely there is high variation between the input bit rate and the output bit rate (from 128 Kbit/s to 32 Kbit/s, for instance) we have *hard transcoding conditions*. In these cases, the skipping of consecutive frames is often unavoidable at the transcoder. However, if many consecutive frames are dropped, the typical jerky effect in the transcoded video sequence is observed by the final user. In addition, in a real time setting, where we try to avoid buffer underflow and overflow, a critical situation, leading to the blocking of the transcoding process, can occur, if it is not faced in a proper way. After skipping many consecutive frames, motion vectors and prediction errors can be very large, and so the frame size can exceed the free buffer space. Thus, if this 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 an earlier frame (even it is not needed), in order to prevent a future frame dropping.

We define $\Gamma = IR/R$ representing the ideal ratio between the number of incoming frames and the number of outgoing frames, if the frames would keep the original size of input bitstreams. So, the transcoder skips frames until the number of consecutive skipped frames is equal to $\Gamma - 1$. Considering the total number of frames in the sequence, at least $1 - 1/\Gamma$ of them will be skipped. Then, our strategy forces the transcoder to skip $\Gamma - 1$ consecutive frames (irrespective of any other feature) in order to prevent a number of consecutive skipped frames 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 that it occurs.

4.4 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 of managing the frame skipping in a randomized way.

As we saw in Section 4.1, 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.

5 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 (indicating the quality of transcoded sequence). We compute the PSNR between the transcoded video sequence

	mobile			foreman			coastguard		
	Frames	PSNR	PSNR2	Frames	PSNR	PSNR2	Frames	PSNR	PSNR2
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

Table 1: Number of transcoded frames, PSNR with freezing, PSNR without freezing (PSNR2) for different video sequences ($IR = 128$, $R = 64$ kbps).

	mobile			foreman			coastguard		
	Frames	PSNR	PSNR2	Frames	PSNR	PSNR2	Frames	PSNR	PSNR2
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 2: Number of transcoded frames, PSNR with freezing, PSNR without freezing (PSNR2) for different video sequences ($IR = 128$, $R = 32$ kbps).

and the video sequence decoded after the front encoder. Two kinds of PSNR measures are considered: in the first one, the PSNR takes into account of transcoded and skipped frames, by replacing these last with their previous ones (freezing). In the second one, PSNR considers transcoded frames only. 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.

We report in Tables 1 and 2 the average PSNR obtained by considering both measures explained above.

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.

In Tables 1 and 2, we show the results of the frame skipping strategies considered in this paper, for "standard" and "hard" transcoding conditions[2]. For the first case, we consider $IR = 128$ kbps and $R = 64$ kbps; for the second one, $IR = 128$ kbps and $R = 32$ kbps.

From our experimental results we deduce that, 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, there are not large differences on the PSNR achieved by different frame skipping strategies (see Figures 1, 2 and 3).

By looking at 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 Table 2 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 Figures 2 and 3, we observe that, in hard transcoding conditions, with the "consecutive" policy, the lowest PSNR values are greater than the lowest values of other policies. This happens since the frames are dropped more uniformly.

6 Conclusions

We implemented four frame skipping strategies in order to improve the quality of the temporal video transcoding in a real-time environment. Disregarding the transmission time, we obtained a real time communication with a maximum admitted delay of 500 ms. 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,

[2]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.

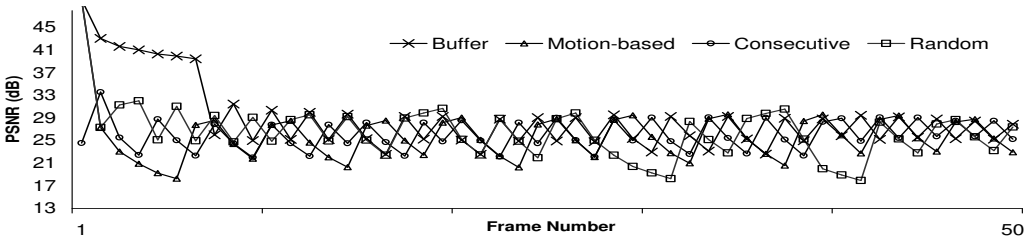

Figure 1: PSNR of the proposed frame skipping policies of "mobile" video sequence ($IR = 128$, $R = 64$ kbps).

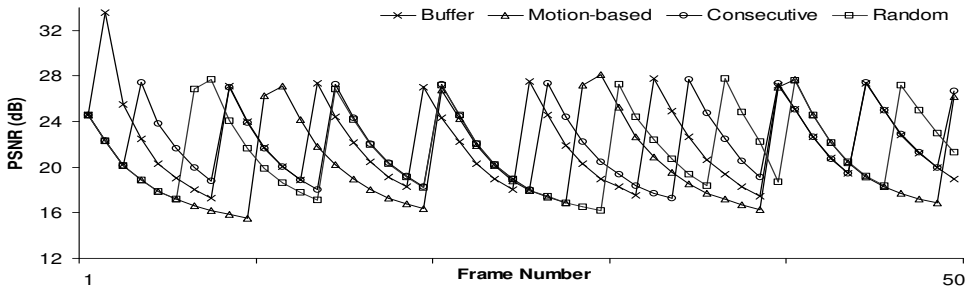

Figure 2: PSNR of the proposed frame skipping policies of "mobile" video sequence ($IR = 128$, $R = 32$ kbps).

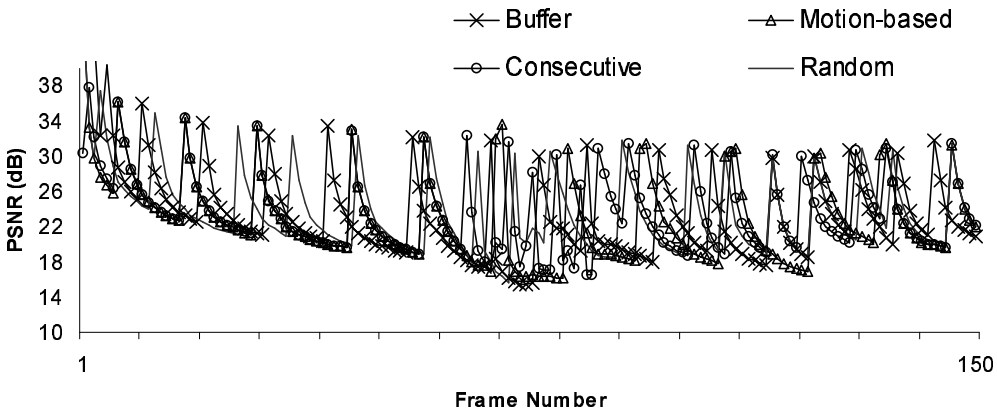

Figure 3: PSNR of the proposed frame skipping policies of "coastguard" video sequence ($IR = 128$, $R = 32$ kbps).

but the "consecutive" policy achieves a better visual quality in hard transcoding conditions, since skipping of an high number of consecutive frames is avoided.

Several problems are still open. An interesting one is an analytical study of the buffer, 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 the emerging H.264-based one.

Acknowledgment

The authors would like to 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 the 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 Transactions on Circuits and Systems for Video Technology*, 12(4):269–275, April 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 *Proceedings of IEEE International Conference on Communications (ICC03)*, volume 2, pages 928–932, Anchorage, Alaska, USA, May 2003.

[3] S. Dogan, S. Eminsoy, A. H. Sadka, and A. M. Kondoz. Video content adaptation using transcoding for enabling UMA over UMTS. In *Proceedings of the 5th International Workshop on Image Analysis for Multimedia Interactive Services (WIAMIS'2004)*, Lisbon, Portugal, April 2004.

[4] 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.

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

[6] K.-T. Fung, Y.-L. Chan, and W.-C. Siu. Low-complexity and high quality frame-skipping transcoder for continuous presence multipoint video conferencing. *IEEE Transactions on Multimedia*, 6(1):31–46, February 2004.

[7] J.-N. Hwang, T.-D. Wu, and C.-W. Lin. Dynamic frame-skipping in video transcoding. In *Proceedings of IEEE Second Workshop on Multimedia Signal Processing*, pages 616–621, Redondo Beach, CA, USA, December 1998.

[8] B. S. Sethi and I. K. Vasudev. Adaptive motion-vector resampling for compressed video downscaling. *IEEE Transactions on Circuits and Systems for Video Technology*, 9(6):929–936, September 1999.

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

[10] G. Shen, B. Zeng, Y.-Q. Zhang, and M. L. Liou. Transcoder with arbitrarily resizing capability. In *Proceedings of IEEE International Symposium on Circuits and Systems (ISCAS 2001)*, volume 5, pages 25–28, Sydney, Australia, May 2001.

[11] H. Shu and L.-P. Chau. Frame-skipping transcoding with motion change consideration. In *Proceedings of IEEE International Symposium on Circuits and Systems (ISCAS2004)*, volume 3, pages 773–776, Vancouver, Canada, May 2004.

[12] J. Xin, M.-T. Sun, K. Chun, and B. S. Choi. Motion re-estimation for HDTV to SDTV transcoding. In *Proceedings of IEEE International Symposium on Circuits and Systems (ISCAS 2002)*, volume 4, pages 715–718, Scottsdale, Arizona, USA, May 2002.

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