A fast skipping policy for H.263 video transcoder[^]

Maurizio A. Bonuccelli[×]
Dipartimento di Informatica,
Università di Pisa, Via Buonarroti 2, Pisa, Italy
E-mail: bonucce@di.unipi.it

Francesca Lonetti*[×]
Dipartimento di Informatica,
Università di Pisa, Via Buonarroti 2, Pisa, Italy
E-mail: lonetti@di.unipi.it
*Corresponding author

Francesca Martelli
Istituto di Scienze e Tecnologie dell'Informazione,
Consiglio Nazionale delle Ricerche, via Moruzzi 1, Pisa, Italy
E-mail: f.martelli@isti.cnr.it

Abstract: Third generation mobile communication systems offer many advanced types of multimedia services, as video streaming, video telephony and video conference. Transcoding is adopted to deliver video content to a broad range of end users with different preferences and bandwidth constraints. Temporal transcoding is one of the solutions to reduce the overall bit rate by dropping some frames of the video sequence. We propose a temporal transcoding architecture with a new frame skipping policy allowing real-time communication. Simulation results show that our temporal transcoding achieves a better performance than a quality one, and the proposed frame skipping strategy is able to strongly reduce the computation time of the transcoding process.

Keywords: video transcoding, frame skipping, video quality.

Biographical notes: Maurizo A. Bonuccelli is a Professor at Computer Science Department of University of Pisa, Italy. He has been associated with that department since 1982, with the only exception the years 1990-1994, during which he was a Professor of computer science at the University of Rome, La Sapienza, Rome, Italy. During 1981 he took a sabbatical leave with IBM T.J. Watson Research Center, Yorktown Heights, NY, working in the computer communications group. He spent September, 1993, at ICSI, Berkeley, CA, with the TENET group. His field of interest is the design and management of communication, computer networks, and distributed parallel processing systems, with a special emphasis on algorithmic and complexity issues. He is interested also in video coding and transcoding.

Francesca Lonetti received the Laurea degree in Computer Science from the University of Pisa in 2003. Since 2004 she is a Ph.D. student at Computer Science Department of University of Pisa. She is involved also in "Video Transcoding for Mobile Telephony" project, at PisaTel Lab of ISTI–CNR, Pisa. Her current research interest is video coding and transcoding for multimedia systems and wireless networks.

Francesca Martelli received the Laurea degree in Computer Science from the University of Pisa in 2000. Since 2001, she is a grant owner at PisaTel Lab of ISTI–CNR, Pisa, within the project "Video Transcoding for Mobile Telephony" in collaboration with Ericsson Lab Italy She is also a Ph.D. student at Computer Science Department of University of Pisa. Her research mainly focuses on packet scheduling algorithms in single-hop multichannel systems; moreover she is interested in mobile ad hoc networking and video transcoding for 3rd generation telephony.

[^] This work has been supported by Ericsson Lab Italy, within the PisaTel Lab at ISTI-CNR

[×] Also at Istituto di Scienze e Tecnologie dell'Informazione, Consiglio Nazionale delle Ricerche, via Moruzzi 1, Pisa, Italy

1 INTRODUCTION

Third generation mobile communication systems (e.g. UMTS) offer new and attractive services 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 networks and services, with their respective quality requirements, is a critical issue. A typical strategy to approach this problem is transcoding, usually performed by servers of a communication system, or by gateways interconnecting different networks.

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 complexity and running time, and enabling the interoperability of heterogeneous multimedia networks. Video transcoding can 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. Spatial transcoding reduces the spatial resolution of a compressed video, it is required for facing the problem of limited size in many access terminals. Quality transcoding reduces the bit-rate of a video sequence, by operating on the bit allocation for each frame and by tuning the quantization parameters. The consequence of this is a variable frame quality. When the bandwidth in a wireless network is very limited, the quality transcoding process, as we will show in this paper, can cause high degradation of the transcoded video quality, if the frame rate is held constant.

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), a 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 its frame rate, without decreasing the quality of not dropped 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. 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, buffer control is a critical issue. Some buffer-based frame skipping policies to allow real time communication have been proposed (Bonuccelli et al., 2005). In this paper, we report our experience in the implementation of a H.263-based temporal transcoder, in comparison to a quality one with TMN8 rate control algorithm. Moreover, we propose a new skipping policy, able to reduce the time of the overall transcoding process.

The paper is organized as follows. In Section 2, we address the temporal transcoding problem, and we survey the results in literature. In Section 3 we present our temporal transcoder architecture able to support real time communication, and in Section 4 we focus on the new skipping strategy. Experimental results are drawn in Section 5. Finally, conclusions and future work are highlighted in Section 6.

2 TEMPORAL TRANSCODING

In temporal transcoding, three issues must be addressed: the motion vectors computation, the prediction errors computation, and the frame skipping policy. The typical strategy adopted in motion vectors computation is Motion Vector Composition (MVC) (Hwang et al., 1998; Shanableh and Ghanbari, 2000; Youn et al., 1999; Chen et al., 2002), together with a restricted motion estimation (RME). 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 one. The new motion vectors are then obtained by searching around the candidate motion vector obtained by MVC, within a few pixels search area. In literature, four MVC algorithms are known: Bilinear Interpolation (Hwang et al., 1998), Telescopic Vector Composition (Shanableh and Ghanbari, 2000), Forward Dominant Vector Selection (Youn et al., 1999), and Activity Dominant Vector Selection (Chen et al., 2002). The new prediction errors are obtained 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 RME). Then, these differences are encoded with the usual DCT (Discrete Cosine Transform) and quantization. An alternative way proposed in (Fung et al., 2002) is to add the errors of the current macroblock to those of the macroblock or reference area in the previous skipped frame. About frame skipping policies, there are in literature several proposals. Many strategies are based on the motion activity value (Hwang et al., 1998; Fung et al., 2002). 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. A weighted definition of this metric has been proposed in (Bonuccelli et al., 2005), where the motion activity is given so that it assumes large value for both frames with few but large motion vectors and with many but small motion vectors. 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 (Shu and Chau, 2004). A different approach has been pursued in (Correia et al., 2003), where a rate control mechanism based on a buffer level prediction algorithm is proposed.

3 OUR TEMPORAL TRANSCODER

In current communication systems, many advanced multimedia applications have real time features. In order to

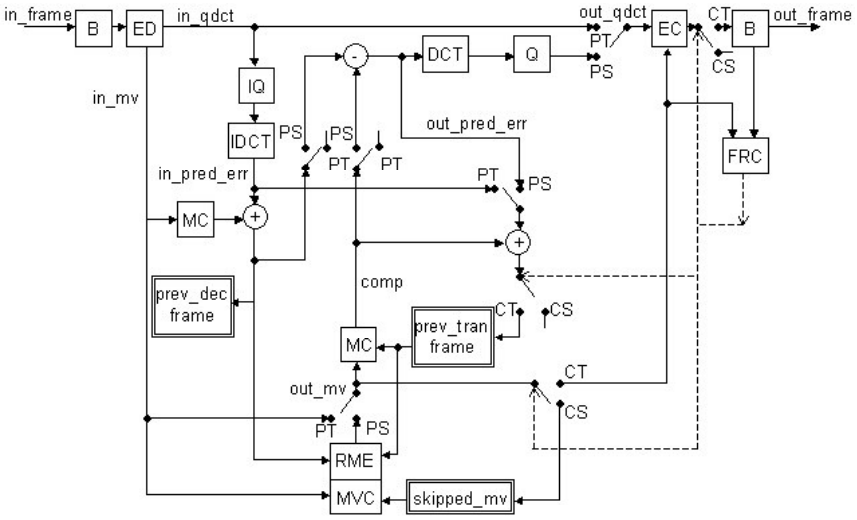
meet the needs of real time applications, we developed a temporal transcoding architecture able to guarantee a fixed communication delay. As in (Bonuccelli et al., 2005), this is done by introducing a transcoder output buffer, that is usually used for facing the real time problem. Before describing our 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 τ=500 ms[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 that reduces 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 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 RME procedure. The prediction errors are computed in the pixel domain. The architecture of our transcoder is shown in Figure 1. The behavior of the transcoder is different according to the reference frame. In other words, at each frame, the transcoder performs some operations if the previous frame has been skipped, and some other operations if the previous frame has been transcoded. The switching between the two behaviors is represented in Figure 1 by the switch “PS/PT”. In addition, also transcoding or skipping of the current frame determines a different behavior (switch “CS/CT”). The left part of Figure 1 depicts the “local decoder” which has to decode every incoming frame *in_frame* by means of motion compensation with the previous decoded frame *prev_dec_frame*. When the reference frame has been transcoded, the only function performed is to store in *prev_tran_frame*, the pixels of the current frame in case that the Frame Rate Control (FRC) module decides to transcode it. Otherwise, if the FRC module decides to skip the current frame, only the motion vectors of the current frame are stored in *skipped_mv*, for being used at the next incoming frame.

In case of skipped reference frame, it is needed to recompute the motion vectors and the prediction errors. The motion vectors are computed by means of motion vector composition (MVC module) and, eventually, restricted motion estimation (RME module). The MVC module adds to the vectors of the incoming frame *in_mv*, the motion vectors chosen among *skipped_mv*, by one of the motion vector composition algorithms, described in Section 2. The RME module performs a resctricted motion estimation around the vectors given by MVC, in order to produce the best possible motion vectors. Then, the motion

Figure 1 *Our temporal transcoder architecture*



[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

compensation is applied to the last not skipped frame *prev_tran_frame* to produce the moto-compensated frame *comp* which is then subtracted from the current decoded frame to produce the prediction errors for the current frame. Finally, if the FRC module decides to transcode the current frame, the prediction errors *out_pred_err* will be added to *comp* in order to store the current frame in *prev_tran_frame*. After the DCT and Q modules, the prediction errors form the current frame together with the motion vectors *out_mv*. Otherwise, if the frame will be skipped *out_mv* will be store in *skipped_mv*. Before applying the skipping policy, it is needed to transcode each frame of the input video sequence, since, as we shall see in the following section, it is needed 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. We propose in 4.1 a new frame skipping policy that avoids the computation of the transcoding process for the frames that will be skipped.

4 FRAME SKIPPING POLICIES

In order to meet the real time constraint, a buffer-based policy is proposed in (Bonuccelli et al., 2005). In that policy, 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*. 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 the buffer-based policy, the buffer overflow is avoided by imposing 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. For more details, see (Bonuccelli et al., 2005).

In the next subsection, we describe a new policy that improves the computation time of the transcoding process of the above policy.

4.1 Size-Prediction Policy

In temporal transcoding, the size of a transcoded frame increases if many previous frames are skipped, that is when the motion vectors and prediction errors of the transcoded frame are obtained by adding those ones of the skipped frames. We observed experimentally that the size of a frame grows according to the logarithm of the number of the previously skipped frame by this law:

$$l(f) = \alpha \ln(f + 1) \quad (1)$$

where *l(f)* is the size of the frame transcoded after skipping f consecutive frames, and α is a constant proportional to the size of the first skipped frame. The size-prediction policy is applied when a frame is skipped. This policy predicts according to (1), the size of the next frame, in order to avoid buffer overflow, if this size is higher than the free buffer space, the frame is skipped. We note that, in our assumptions, buffer occupancy decreases at a constant rate of R/ρ bits every $1/\rho$ seconds. The frame is transcoded only when its predicted size is lower than the free buffer space. However, as in the buffer-based policy of (Bonuccelli et al., 2005), in order to avoid buffer underflow a frame is transcoded if the buffer occupancy is lower than a properly tuned threshold. Compared with the buffer-based policy mentioned in (Bonuccelli et al., 2005), this one has the advantage of predicting the size of a frame avoiding the computation needed to transcode it, and greatly reducing the time of the total transcoding process when many consecutive frames are skipped. As we will show in Section 5.2, the performance of this policy is comparable to that of the buffer-based one. The pseudo-code of size-prediction policy is shown in Figure 2.

Figure 2 *The size-prediction policy pseudo-code*

Size-Prediction Policy (frame f):
  If *(f = first frame)* **then** *transcode f*
    Else
      If *($(L \le B_{lower}(S))$ & $(L + l(f) \le S)$)* **then** *transcode f*
        Else If $(L + l(f) > S)$
          Do
            skip frame f
            predict the size of frame f + 1
            f = f + 1
            $L = L - R/\rho$
          while *($(L > B_{lower}(S))$ & $(L + l(f) \ge S)$)*
          transcode f

5 SIMULATION RESULTS

We implemented an H.263-based transcoder and evaluated its performance with respect to a quality transcoder. We show the results in Section 5.1. Moreover, we evaluated a size-prediction policy with respect to the buffer-based policy of (Bonuccelli et al., 2005): results are described in Section 5.2.

We considered three metrics: number of transcoded frames (indicating the video sequence smoothness), PSNR (indicating the quality of a video sequence by taking into account the differences of luminance values of corresponding pixels in the original and reconstructed frame), and total processing time.

We computed the PSNR in this way: we considered as original video sequence, that one decoded after the front encoder. As reconstructed sequence, we used that obtained after the transcoding, where skipped frames are replaced

with their previous ones (freezing). This way (that here we call PSNR1) of computing the PSNR allows us to measure the actual visual quality perceived by the final user. Another way (that we call PSNR2) is to consider only transcoded frames, so measuring the quality of single frames, without capturing the degradation introduced by frame dropping.

We considered several video sequences of 300 frames in QCIF format and frame rate of 30 fps. In the following we show only the most significant experimental results about two different bit-rate reductions: the first with high bit-rates, i.e. from *IR*=256 Kbps and *R*=128 Kbps, and the latter with low bit-rates, i.e. *IR*=64 Kbps and *R*=32 Kbps.

5.1 Temporal vs Quality transcoder

We implemented a quality transcoder which decodes the incoming video sequence at bit rate *IR*, and re-encodes it with bit rate *R*, by using the same rate control algorithm of the front encoder (TMN8). We compared it with our temporal transcoder described in Section 3. As shown in Table 1, our temporal transcoder has a comparable computation time than the quality one, but obviously skips more frames, so it produces a sequence with lower degradation of quality transcoder. For getting an idea of this degradation, look at Figure 3, were we report a frame of foreman video sequence encoded at *IR*=64 Kbps (on the left) and quality transcoded to *R*=32 Kbps (on the right). We perceive a quality degradation introduced by the quality transcoding. So, comparable values of average PSNR do not correspond to the same visual quality. This is shown also in Figure 4, where the PSNR1 values of the quality transcoded frames are quite constant, while the values of temporal transcoded frames are higher or lower depending whether the frame has been transcoded or skipped, respectively.

5.2 Size-prediction policy evaluation

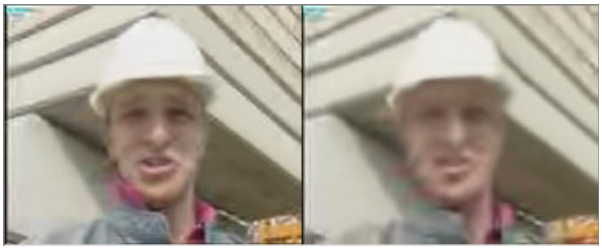
We evaluated size-prediction policy, by comparing it with buffer-based policy of (Bonuccelli et al., 2005). We report in Table 2, the number of transcoded frames, the average PSNR1 and PSNR2, and the total time of the transcoding process. Notice that the two policies have almost the same performance in terms of number of transcoded frames and PSNR values, but the computation time of size-prediction policy is much lower (with a decrease of 30-45%). Besides, as we can see in Figure 5 (*IR*=64 Kbps and *R*=32Kbps), the two policies follow the same behavior, in terms of PSNR1 values: in fact the two plots are very similar. The same behavior is obtained with *IR*=256 Kbps and *R*=128 Kbps.

Table 1 *Temporal vs Quality Transcoding*

	Temporal transcoder			*Quality Transcoder*		
	# Frames	*PSNR1 (dB)*	*Time (sec)*	*# Frames*	*PSNR1 (dB)*	*Time (sec)*
IR=256 kbps, R=128 kbps						
Akiyo	96	43,48	9,1	300	39,64	9,6
Mobile	161	30,18	8,2	298	27,02	9,9
Foreman	110	32,38	9,4	299	34,77	9,7
Coastguard	118	32,39	9,5	300	32,47	9,3
IR=64 kbps, R=32 kbps						
Akiyo	104	40,10	7,2	291	35,57	7,3
Mobile	142	26,60	6,6	156	25,97	6,9
Foreman	112	29,88	7,7	218	28,76	7,1
Coastguard	144	31,36	7,1	285	28,79	7,8

smoothness. On the other hand, temporal transcoded frames have an higher quality, which it is the same of the front encoder. As shown in Table 1, the average PSNR1 values are greater than those of quality transcoder in most cases, especially at low bit-rates, where it is more evident the

Figure 3 *A frame of foreman video sequence coded with IR=64 (on the left) and quality transcoded with R=32 Kbps (on the right)*



6 CONCLUSIONS

We implemented an H.263-based temporal transcoder and a skipping strategy allowing real-time communication. We compared our temporal transcoder with a quality one and we concluded that the temporal one has better performance than the quality one especially at low bit rates. We also developed a new frame skipping policy that allows to speed up the transcoding process by avoiding any computation when frames are skipped. Besides, this policy guarantees real time communication. Simulations showed that this

Figure 4 *PSNR1 of foreman video sequence temporal and quality transcoded (IR=256 Kbps, R=128 Kbps)*

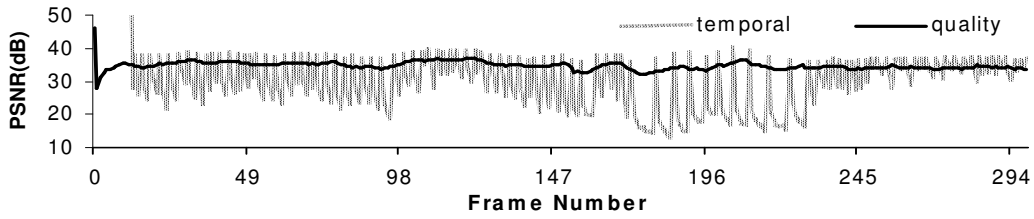
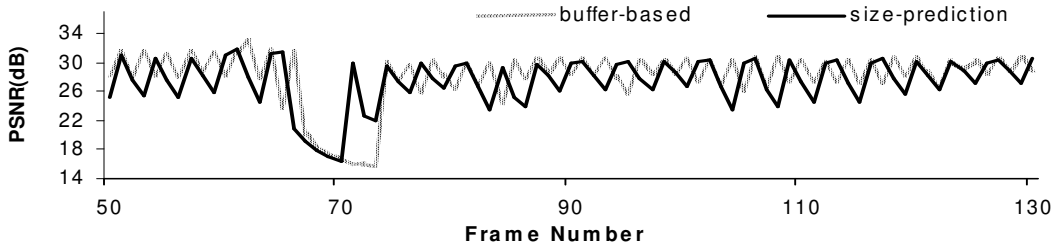


Table 2 *Buffer-based vs. Size-prediction frame skipping policy*

	Buffer-based				*Size-prediction*			
	#Frames	*PSNR1 (dB)*	*PSNR2 (dB)*	*Time (sec)*	*#Frames*	*PSNR1 (dB)*	*PSNR2 (dB)*	*Time (sec)*
IR=256 kbps, R=128 kbps								
Akiyo	96	43,48	52,56	9,1	94	43,02	52,22	6,1
Mobile	161	30,18	34,30	8,2	154	29,29	34,39	5,3
Foreman	110	32,38	44,49	9,4	110	32,23	44,29	6,5
Coastguard	118	32,39	41,69	9,5	113	31,82	41,57	6,1
IR=64 kbps, R=32 kbps								
Akiyo	104	40,10	44,62	7,2	102	39,64	44,30	4,1
Mobile	142	26,60	27,81	6,6	127	25,75	27,79	4,1
Foreman	112	29,88	36,10	7,7	109	29,29	36,45	4,6
Coastguard	144	31,36	34,82	7,1	129	30,60	34,93	4,4

Figure 5 *PSNR1 of coastguard video sequence with buffer-based and size-prediction policies (IR=256 Kbps, R=128 Kbps)*



policy performs as good as the buffer-based proposed in (Bonuccelli et al., 2005). We plan to develop an hybrid temporal/quality transcoding architecture and to test the behavior of our temporal transcoding architecture by implementing it with the emerging H.264 codec.

ACKNOWLEDGEMENT

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. We thank also Luca Leonardi and Gianni Rosa for their contribution to this work as part of their Master Theses.

REFERENCES

Bonuccelli, M.A., Lonetti, F. and Martelli, F. (2005) 'Temporal Transcoding for Mobile Video Communication', To appear in the *Proceedings of the 2th International Conference on Mobile and Ubiquitous System: Networking and Services*, San Diego, CA, USA. Available at: http//www.di.unipi.it/~f.martel/publications.html

Hwang, J.N., Wu, T.D. and Lin, C.W. (1998) 'Dynamic frame-skipping in video transcoding', *Proceedings of 2nd Workshop on Multimedia Signal Processing*, Redondo Beach, CA, USA, December, pp.616–621.

Shanableh, T. and Ghanbari, M. (2000) 'Heterogeneous video transcoding to lower spatio-temporal resolutions and different encoding formats', *IEEE Transactions on Multimedia*, Vol. 2, No. 2, pp.101–110.

Youn, J., Sun, M.T. and Lin, C.W. (1999) 'Motion vector refinement for high-performance transcoding', *IEEE Transactions on Multimedia*, Vol. 1, No. 1, pp.30–40.

Chen, M.J., Chu, M.C. and Pan, C.W. (2002) 'Efficient motion-estimation algorithm for reduced frame-rate video transcoder', *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 12, No. 4, pp.269–275.

Fung, K.T., Chan, Y.L. and Siu, W.C. (2002) 'New architecture for dynamic frame-skipping transcoder', *IEEE Transactions on Image Processing*, Vol. 11, No. 8, pp.886–900.

Shu, H. and Chau, L.P. (2004) 'Frame-skipping Transcoding with motion change consideration', *Proceedings of International Symposium on Circuits and Systems*, Vancouver, Canada, May, pp. 773–776.

Correia, P.D.F., Silva, V. and Assunção, P.A. (2003) 'A method for improving the quality of mobile video under hard transcoding conditions', *Proceedings of International Conference on Communications,* Anchorage, Alaska, USA, May, pp. 928–932.