Project Report
(January 2003 - December 2005)

VIDEO TRANSCODING TECHNIQUES IN MOBILE SYSTEMS

COORDINATOR:

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

PARTICIPANTS:

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

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

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

Contents

Chapter 1

1.1 Executive summary

The third generation telecommunication systems (UMTS) will provide more advanced types of interactive and distribution services, and actually digital video is one of the most prominent applications for multimedia communications. The key technology that enables many applications such as Digital TV broadcasting, Distance Learning, Video on Demand, Video Telephony and Videoconferencing, is digital video coding. In third generation telecommunication systems, communication technologies are highly heterogeneous. Adapting the media content to different network 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. Transcoding is the process of converting a video into another one with different features, in order to adapt the video content to different network features, like channel bandwidth, terminal capabilities and user preferences.

The goal of this project is to develop efficient solutions for achieving an optimal video quality in real-time video transcoding. The project started in January 2003. The initial step has been to study different video codec standards, first MPEG4 and H.263 and later the emergent H.264. Different transcoding techniques were investigated, with particular focus on temporal transcoding process. Temporal transcoding consists in dropping some frames from the video sequence, and correctly reconstructing the non-dropped ones. The main issue has been to reduce the computational complexity for performing temporal transcoding, by looking for a trade-off between compression and computation time of block matching techniques. However, real-time applications are delay sensitive, and so another goal of this project was to look for a trade-off between transcoding performance (in terms of video quality and computation time) and delay of data delivery. This was done by proposing different buffer-based frame skipping policies. Temporal and quality transcoding were compared. The project terminated in December 2005. The following publications have been produced during this project:

M.A. Bonuccelli, F. Lonetti, F. Martelli, "***Temporal Transcoding for Mobile Video Communication***", Proceedings of 2nd International Conference on Mobile and Ubiquitous System: Networking and Services (MobiQuitous 2005), San Diego, CA, USA, July 17-21, 2005.

M.A. Bonuccelli, F. Lonetti, F. Martelli, "***Video Transcoding Architectures for Multimedia Real Time Services***", ERCIM News No. 62, July 2005, p. 39-40.

M.A. Bonuccelli, F. Lonetti, F. Martelli, "***A fast skipping policy for H.263 video transcoder***", Proceedings of 12th International Workshop on Systems, Signals & Image Processing, (IWSSIP '05), Chalkida, Greece, September 22-24, 2005.

1.2 Scope & context

Several video transcoding schemes have been developed in the last years, which provide coding standard conversion, bit rate conversion (quality transcoding), resolution scaling (spatial transcoding), and frame rate conversion (temporal transcoding).

We are interested in temporal transcoding schemes. We developed two temporal transcoding architectures called DFS (Dynamic Frame Skipping) and FSC (Frame Skipping Control). We evaluated them realizing that DFS architecture is better than FSC architecture. We evaluated DFS architecture with different motion vector composition algorithms, realizing that their performances are very similar. We compared our temporal transcoding with a quality one for different video sequences concluding that temporal transcoding is better for video sequences with little motion, while quality transcoding is better for video sequences with a lot of motion. To consider this motion information in transcoding process, we developed motion based frame skipping policies. An important issue in development of skipping policies, is output transcoder buffer occupancy, that determinates the delay of data delivery. We proposed five buffer-based skipping policies. Among them, the “consecutive” frame skipping policy is used in hard transcoding conditions, that is when an high variation between input and output bit rate occurs.

We evaluated the performances of different frame skipping policies with MPEG4 and H.263 codecs. We implemented our temporal transcoding architecture with H.264 codec. For the variable partition of H.264 macroblocks, we needed to develop a new motion vector composition scheme. We implemented also a new rate control algorithm to improve the video quality of encoded frames, which influences that one of the transcoded video sequence.

1.3 Structure

This report is organized as follows. In Chapter 2, we give a brief overview of video coding features. In Chapter 3, we discuss video transcoding techniques. In Chapter 4, we focus on temporal video transcoding issues presented in literature. We present our research results in Chapter 5, and future work in Chapter 6. In Appendix A, there is a schematic presentation of this report.

Chapter 2: Video coding features

In the last years, many video coding standards have been proposed for various video applications, such as H.263 for low-bit rate video communications, MPEG-1 for storage media applications, MPEG-2 for broadcasting and general high quality video application, MPEG-4 for streaming video and interactive multimedia applications, H.264 for high compression request. They are all based on the same framework: hybrid DCT (Discrete Cosine Transform) and MCP (Motion Compensated Prediction) coding. The video sequences are composed by frames or images captured at regular time intervals. Each image is a two-dimensional matrix of points, named pixels. Images usually are stored in the 24-bit RGB (Red, Green and Blue) format with 24 bits for each pixel, 8 bits for each colour channel. A more efficient format is 24-bit YC_bC_r: 8 bits for the luminance signal (Y) representing the black and white image, and 8 bits for each of the two chrominance signals (C_b and C_r), that represent colour information. The chrominance signals can be sub-sampled with a 2:1 compression in both horizontal and vertical dimensions, reducing the 24 bits per pixel to 12-bits per pixel. This sub-sampling does not affect the perceived video quality since the human eye is less sensitive to colour variations than to luminance variations [36]. Each video frame to be coded is divided into non-overlapping macroblocks. The number of blocks composing a macroblock depends on the given format. In many coding formats, each macroblock consists of four 8*8 luminance blocks and two corresponding 8*8 chrominance blocks (one C_b block and one C_r block).

There are three types of video coding: intra-frame coding (I-frame), forward predicted frame coding (P-frame), and bi-directional predicted frame coding (B-frame).

An I-frame is encoded independently without referring to other frames. Each block in an I-frame is first DCT-transformed into the frequency domain. Typically, the low frequency coefficients reside on the top left corner, while the high frequency coefficients reside on the bottom right corner. This fact is very useful because it is simple to characterize the information that can be eliminated: in fact, the human eye is more sensitive to low spatial frequencies than to high spatial ones. Then, a little distortion introduced on the high frequencies coding does not change the perception of the image. After DCT, the coefficient values are quantized. The quantization, that is the division by an integer positive value with rounding, introduces the largest and non recoverable error. By varying the quantization parameters, it is possible to set the compression factor, and consequently the video quality. The resulting quantized DCT coefficients are then entropy coded.

The entropic coding is developed in three steps: a zig-zag scan of the block's coefficients in order to obtain a sequence with long subsequences of zeros; the run length coding, which codes the consecutive repetitions of the same symbol with a single occurrence, followed by a counter indicating the number of repetitions; finally, the variable length coding (VLC), assigning to the most statistically frequent symbol a shortest representation.

A P-frame is encoded relatively to its past reference frame. A reference frame can be a P-frame or an I-frame. A macroblock in a P-frame may be encoded as an intra-macroblock or an inter-macroblock. An intra-macroblock is encoded like a macroblock in an I-frame. An inter-macroblock is encoded as a motion vector to 16*16 area in the past reference frame, plus the corresponding differences (prediction errors) between the area and the current macroblock. Such differences are coded with intraframe coding.

The search of the motion vector, giving the smallest prediction errors, is named Motion Estimation procedure (ME) and it is the most computationally intensive part of the video coding process.

A B-frame is encoded relative to its past reference frame and future reference frame. The encoding of a B-frame is equal to a P-frame, except that the motion vectors may refer to areas in the future reference frame.

A typical coded video sequence includes all three types of frames and a fixed frame type mixing pattern is repeated throughout the entire video stream. An example of MPEG video frame type mixing pattern, named GOP (Group of Pictures) is shown in Figure 1, where the arrows indicate the prediction directions.

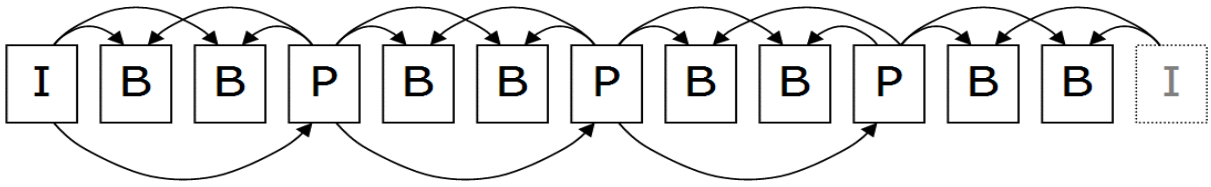


Figure 1. GOP (Group of Pictures)

A block diagram of a typical hybrid DCT/MCP video codec is shown in Figure 2. The encoder on the left in the figure generates a variable bit-rate bit stream. The encoder buffer is needed to buffer the bit-stream when the instantaneous output bit-rate is higher than the channel capacity. The control of the output bit rate according to the channel bandwidth is the subject of video rate control. The decoder on the right in the figure performs simply a reverse process of the encoding. In Figure

2, we note that the encoder contains a so-called local decoder, which is identical to the remote decoder. This local decoder produces an exact replica of the video frame at the remote decoder's output. This local decoding operation is necessary, since the previous original frame is not available at the remote decoder that uses the reconstructed version of the previous frame, in order to produce the current frame. This measure ensures that the decoder uses the same reconstructed frame as the one used by the encoder to make the motion compensation.

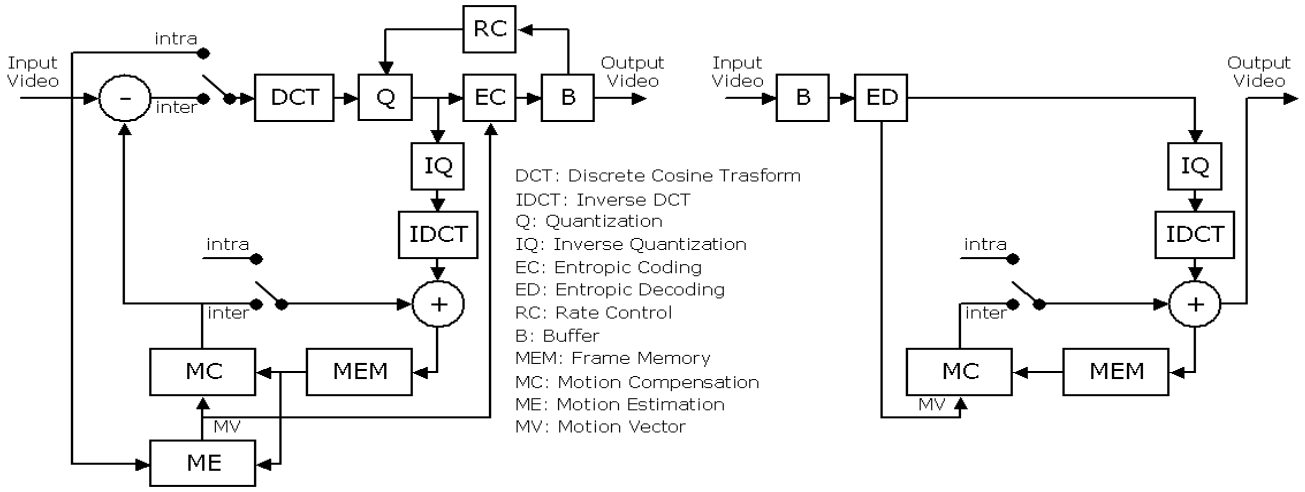


Figure 2. A hybrid MCP/DCT video codec

Different coding standard present advanced coding features. H.263 presents half-pixel motion compensation and improved VLC coding. MPEG-4 is similar to H.263 but supports several profiles to address different applications. The Simple Profile (SP) is targeted at low-bit rate, low delay video communications. The Advanced Simple Profile (ASP) includes 1/4 pixel motion compensation and Bidirectional Frames. The latest video coding standard is MPEG-4 AVC/H.264 and it reflects the latest advances of video coding techniques. The major improvements over H.263 and previous MPEG-4 video coding profiles are: multiple reference frames, variable block-sizes, 4*4 integer DCT-like transform, improved intra prediction, arithmetic coding, 1/8 pixel interpolation. All these video coding standards only standardize the decoder, not the encoder. This allows the encoder performance to be further improved by technology advancements without affecting the interoperability. The technologies making the most difference in the encoder performance include the motion estimation and the rate control.

2.1 Motion Estimation

Motion estimation is the technique to estimate the motion vectors in a video sequence. In most existing video coding standards, the motion estimation is block based. Given a reference frame and an M*N block in the current frame, the objective of motion estimation is to find the best-matched M*N block in the reference frame within a search region relative to the position of current block, as illustrated in Figure 3.

The minimum SAD (Sum of Absolute Difference) is the most commonly used matching criterion for this choice. The SAD is defined as:

$$SAD\ (i,j) = \sum_{K=0}^{M-1} \sum_{l=0}^{N-1} \left| Pc(x+k, y+l) - \Pr(x+i+k, y+j+l) \right| \quad (2.1)$$

where $Pc(x+k, y+1)$ is the luminance pixel of the block in the current frame, $P_r(x+i+k, y+j+l)$ is the luminance pixel in the reference frame, $-p <= i <= p$, $-p <= j <= p$, and p determines the search range.

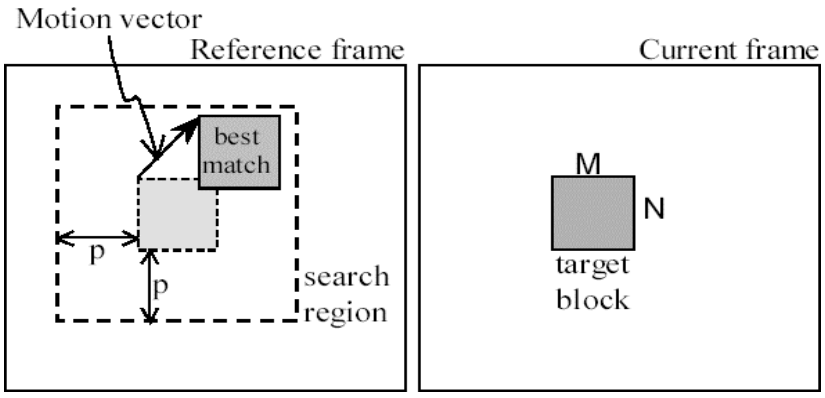


Figure 3. Block Motion Estimation

The exhaustive search (full-search) algorithm that searches every possible candidate block in the search range, gives the best performance. However, it is not suitable for many practical applications due to its high computational complexity. Various fast motion-estimation algorithms have been developed, which trade off estimation accuracy to reduce the

computation. Three-Step Search [22] is one of the most popular fast motion-estimation algorithms. As shown in Figure 4, in each step, nine search-points are checked. After each step, the step-size is reduced by half, and the search ends with a step-size of one pixel. At each new step, the search centre is moved to the best matching point from the previous step.

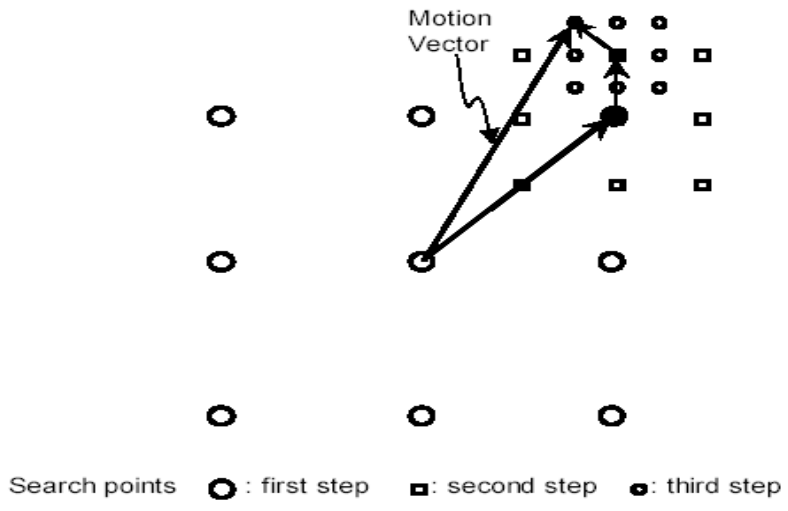


Figure 4. Three-step search method

2.2 Rate Control

The rate control scheme operating on quantization parameters determines the video quality. Video quality is a subjective measure. The metric adopted in literature to compute the video quality is PSNR (Peak Signal to Noise Ratio) measure. The most commonly used formula for computing the PSNR is:

$$\text{PSNR} = 10 \log_{10}\left(\frac{255^2}{MSE}\right) \quad (2.2)$$

The formula to compute MSE (Mean Square Error) is:

$$\text{MSE} = \frac{\sum_{i=0}^{H-1}\sum_{j=0}^{W-1}\left[I_k(\mathrm{i},\mathrm{j}) - \hat{I}_k(\mathrm{i},\mathrm{j})\right]^2}{W \times H} \quad (2.3)$$

where *W* and *H* are the horizontal and vertical dimensions of the frame, $I_k(i,j)$ and $\hat{I}_k(i,j)$ are the luminance intensity of the (i,j) pixel of frame k in the original and in the reconstructed frame, respectively.

In order to transmit the video over a constant-bit-rate channel, such as the PSTN (Public Switched Telephone Network) or DSL lines, the encoder buffer is used to buffer the bit-stream when the instantaneous output bit-rate is higher than the channel capacity. To prevent the encoder buffer from overflow or underflow, a rate-control algorithm is used to adjust the quantization parameter in order to control the produced bit-rate. A variable quantization parameter causes a variable Peak-Signal-to-Noise-Ratio (PSNR) and a constant output encoder buffer bit rate that is the same rate as the communication channel.

The challenge of rate-control in video encoding is to determine the quantization parameter for video frames in order to achieve the best video quality given the application constraints (channel bandwidth, delay, etc.). Currently, rate-control is still an active research area.

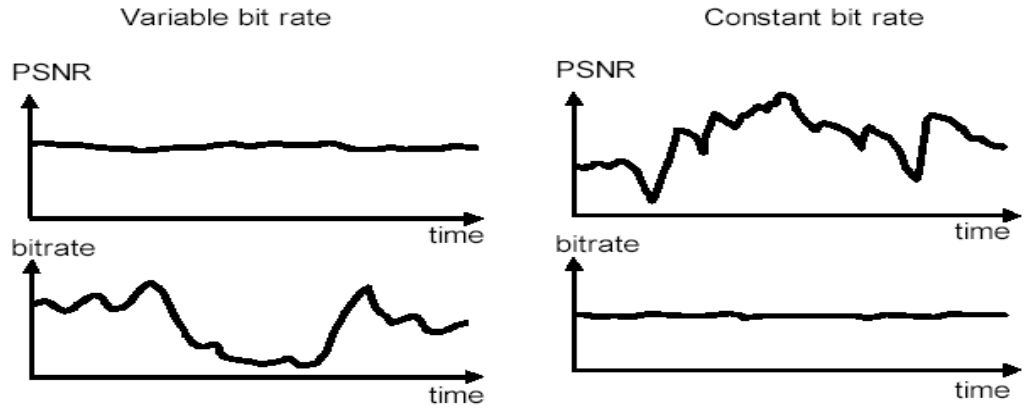


Figure 5. Variable bit-rate vs. constant bit-rate

Chapter 3: Video transcoding techniques

In 3G communication systems, the amount of multimedia content that is transmitted over optical, wireless and wired networks is growing. Furthermore, in each of these networks, there are a variety of multimedia terminals that support different formats. This scenario is often referred to as Universal Multimedia Access (UMA) and is illustrated in Figure 6. The involved networks are often characterized by different network bandwidth constraints, and the terminals themselves vary in display capabilities, processing power and memory capacity. The goal of video transcoding is to represent and deliver the content according to the current network and terminal characteristics. Input and output of a video transcoder are typically compressed video streams that conform to certain compression standards. The decoder decodes the input video into the pixel-domain, and then the encoder encodes the decoded frames into the desired compressed video formats. Different from conventional video coding, video transcoding knows not only the decoded video frames, but also the coding statistics of the input compressed video. This information can be exploited to perform the video transcoding process.

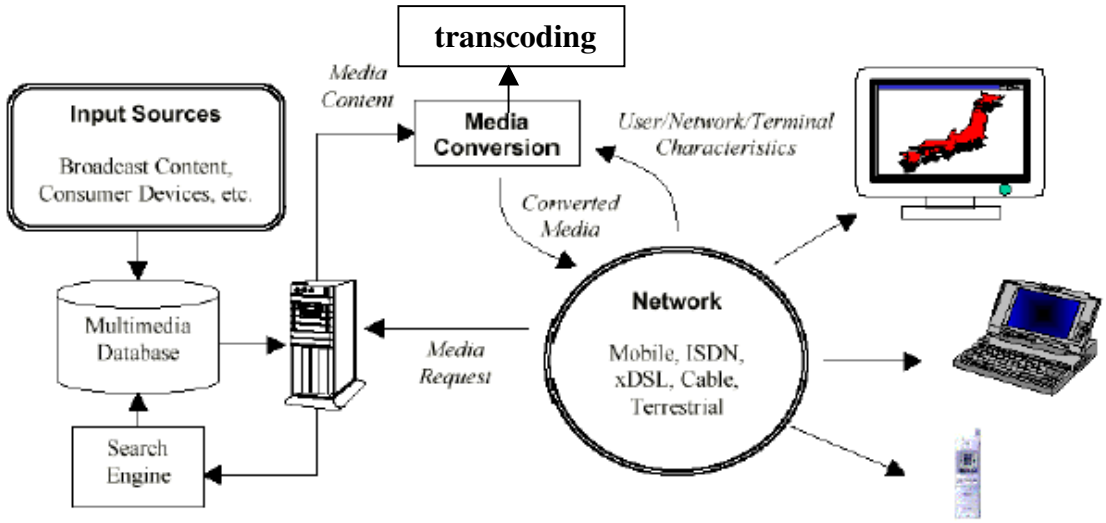


Figure 6. Universal Multimedia Access (UMA): conversion of multimedia content

3.1 Video transcoding architectures

Several typical transcoding architectures have been proposed in the literature. They are grouped into three categories: Pixel Domain Transcoder, DCT Domain Transcoder and Open-Loop Transcoder. The most straightforward structure for video transcoding is a cascaded pixel domain video transcoder (CPDT), which connects a standard decoder with a standard encoder together as illustrated in Figure 7. However, the computational complexity of this approach is very high since it requires performing both encoding and decoding.

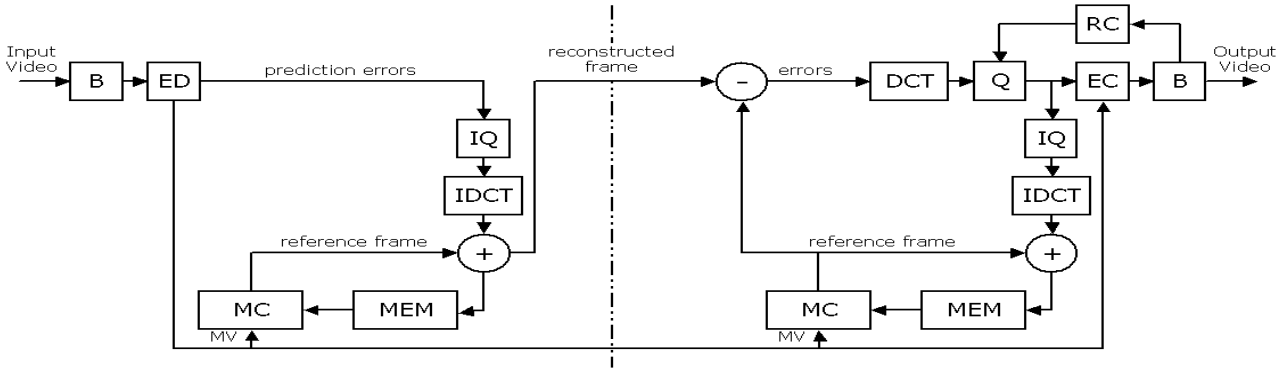


Figure 7. Cascaded Pixel Domain Transcoder

The challenge in the transcoding research is how to achieve the optimal trade-off between computational complexity and video quality.

In CPDT, decoded motion vectors can be reused so that the motion estimation can be avoided. A simplified architecture, named Fast Pixel Domain Transcoder (FPDT), is derived and showed in Figure 8.

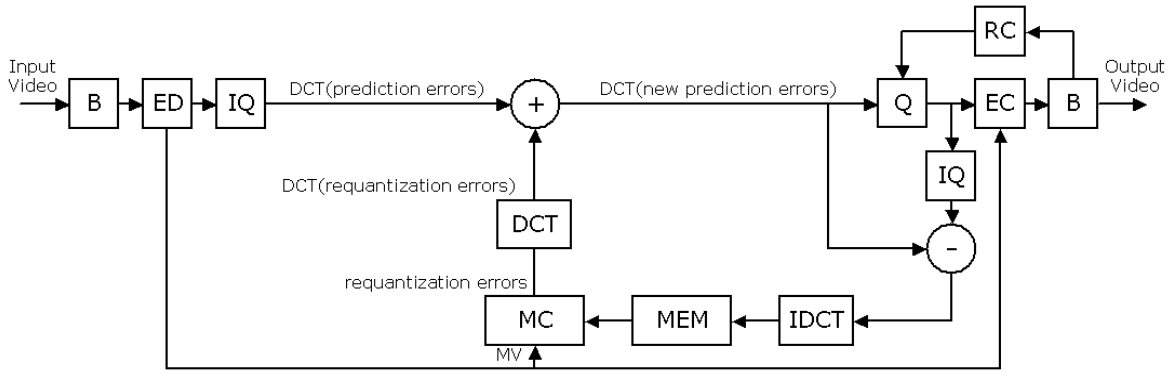


Figure 8. Fast Pixel Domain Transcoder

In this simplified architecture, the whole transcoding process is performed in the DCT domain except the motion compensation loop. The most time-consuming parts are in the DCT and IDCT modules. These two modules can be removed from the architecture performing the motion compensation in the DCT domain, by several DCT domain interpolation algorithms [23][24]. This transcoding architecture is showed in Figure 9.

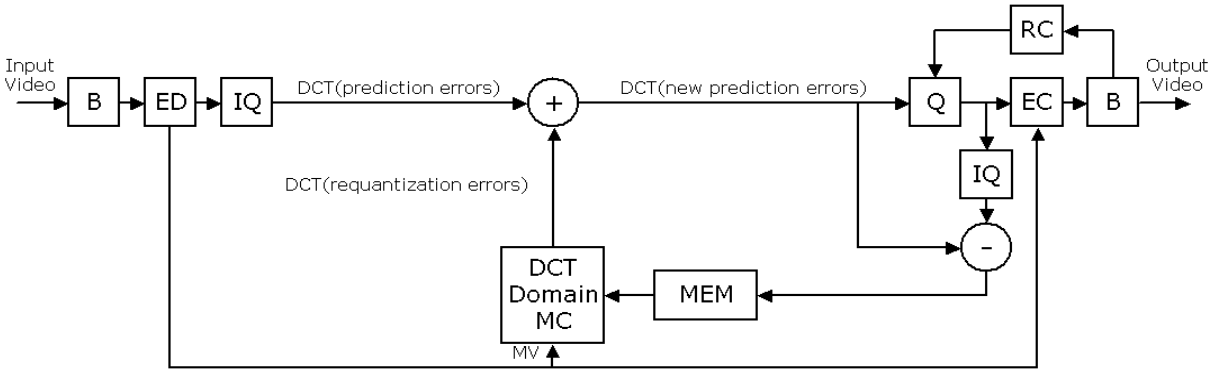


Figure 9. DCT Domain Transcoder

For fast scaling video bit rates, some open-loop transcoding architectures have been proposed in the literature [25][26]. A typical open-loop transcoder is illustrated in Figure 10. In this architecture, the incoming video stream is variable-length decoded and dequantized without further decoding. Two approaches are used for bit rate scaling in the open-loop transcoder: cutting the high frequency DCT coefficients and increasing the quantization step. In both approaches, the motion vectors, coding mode and other syntax elements from the incoming video stream are reused by the transcoder.

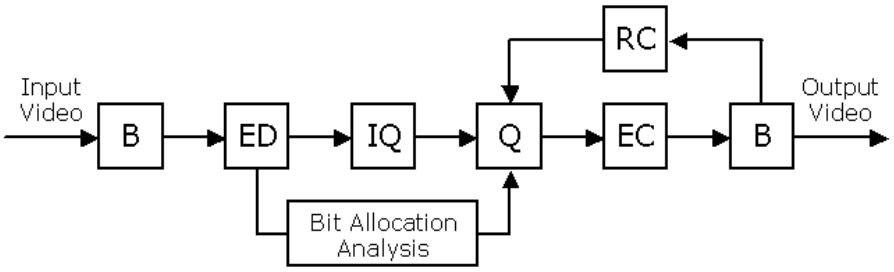


Figure 10. Open Loop Transcoder

From the above introduction of transcoding architectures, we can easily see that the pixel domain transcoder is the most complicated one in terms of overall structure, while the open loop transcoder is the simplest one. However, in open loop transcoding, the visual quality degrades for dropping high frequency coefficients and for the requantization error. At the same time, a drift error is caused by the loss of high frequency data, which creates a mismatch between the actual reference frame used for prediction in the transcoder and decoder. Since the current reconstructed picture is also used for future predictions, the drift error propagates to future frames, and thus the distortion of video quality increases. Pixel domain transcoder architectures (CPDT) usually have better visual quality than DCT domain transcoders and Open Loop transcoders. However, if the motion vectors from the incoming video stream are reused for transcoding in the Fast Pixel domain transcoder, a significant part of computation is reduced because more than 70% of the total computation of an encoder is due to the motion estimation module. From this point of view, reusing motion vectors is the most significant step to reduce transcoder complexity. There are some other methods which have been proposed in literature to reduce the complexity of different modules, such as DCT/IDCT, motion compensation, etc., in different transcoding architectures. However, compared to motion estimation, these modules are not computationally expensive. In addition, there is always a trade-off between visual quality and complexity. In most cases, the visual quality will be degraded by simplifying some modules in the transcoder. Some other related research work about transcoding architectures can be found in [27]. Therefore, the choice of architecture and type of transcoding depends on the dedicated application's and user's requirements. The main goal is to avoid cascaded decoding/re-encoding processes, maintaining the quality of service (QoS), reducing the processing power and most significantly the time delay associated with the conventional cascaded decoding/re-encoding processes, most important in delay sensitive applications such as two-way video communications.

A type of transcoding is the format conversion that is needed in order to achieve the interoperability among different coding standards in heterogeneous multimedia networks. This format conversion process operates a syntax change from one video coding standard to another one. In [2], an efficient MPEG-4/H.263 video transcoder is presented. The other types of transcoding are presented in the next sections.

3.2 Spatial Video Transcoding

Video spatial resolution downscaling is important since most mobile devices are characterized by limited screen sizes. For transcoding a compressed video stream with *M*N* spatial resolution into a stream with smaller spatial resolution, such as *M/2*N/2* or *M/4*N/4*, motion vectors from the incoming video can not be reused directly, but they have to be resampled and downscaled. Based on the new motion vectors, prediction errors are recomputed and compressed.

The most typical sample case is the 2:1 downscaling, where four 8*8 DCT blocks are downscaled into one 8*8 DCT block. A simple method that masks the high-frequency DCT coefficients and retains the lowest 4*4 DCT coefficients may be used. However, more sophisticated filtering methods that transform a 16*16 DCT block directly to an 8*8 DCT block have been proposed in [28] .

Several strategies have been proposed to compose the motion vector of the target macroblock using the motion vectors of the input macroblocks, such as random selection, the median, the average and weighted average [10][11][12]. The median method achieves the best performance.

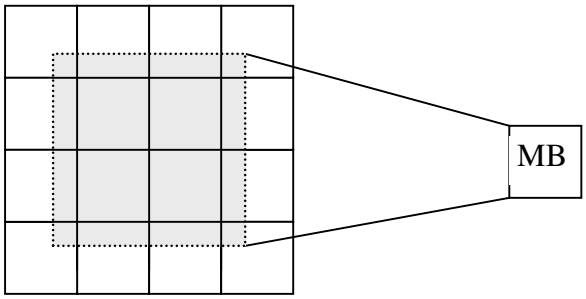


Figure 11. Arbitrary down-sampling spatial transcoding

Recent works extend the previous strategies to the spatial transcoding with arbitrary down-sampling ratio (Figure 11). In this case, due to the non-integer-factor spatial down-sampling, the important issue is how to combine the motion vectors with different contributions in order to choose the motion vector of the target macroblock [13][14].

3.3 Quality Video Transcoding

In order to distribute the same encoded video sequence to several users through channels with different capabilities, the compressed video sequence has to be converted into specific bit rates for each outgoing channel. In most bit adaptation cases, pre-encoded video with high bit rate and fine visual quality needs to be converted into low bit rate video with gracefully degraded visual quality. The ideal result is to achieve visual quality as good as the quality of a video stream that is compressed by an encoder at the low bit rate. All video coding standards assume that the compressed video will be transmitted over a Constant Bit Rate (CBR) channel and the rate control scheme of these standards is based on such assumption. However, this is not true in the real case, since the channel bit-rate is influenced by network congestion, packet loss, high Bit Error Rate (BER) and channel fading effects in wireless links. Many techniques of error recovery and error resilience are proposed in order to solve these problems. Rate control for transcoding differs from encoder rate control in the following ways: the video transcoder reuses the picture types of the incoming video, future picture types of each GOP are, in general, unknown and so they are not used to set the target number of bits for each frame, the relationship between quantizer step-size and bit rate of the incoming video is known, this information is used for setting the target bit rate and quantizer step-size of a frame or macroblock.

Many quality transcoding algorithms achieve the target bit rate reduction by operating on the bit allocation for each frame (Frame-Layer bit allocation) and on the quantization parameters of every macroblock of the frame (Macroblock-Layer rate control), according to the target bit-rate. The Frame-Layer bit allocation strategies are based on the input video sequence complexity, on the type of the frames and on the image distortion [3][4]. The Macroblock Layer rate control techniques determine the quantization parameters for every macroblock of the frame according to the allocated number of bits for each frame. These techniques are based on Lagrangian optimisation [5][6][7], Dynamic Programming [8], or on the relationship between the number of VLC code words in a frame and the produced bits for encoding these VLC code words [9].

3.4 Temporal Video Transcoding

In order to transcode an incoming compressed video bitstream for a low bandwidth outgoing channel, such as a wireless network, a high transcoding ratio is often required. However, high transcoding ratios may result in unacceptable video quality when the incoming bitstream is

transcoded with the full frame rate as the incoming bitstream. For example, in a wireless network which normally has a less than 20 kbps bandwidth, the quality degradation due to the low bit-rate is significant with 25 or 30 frames per second.

Temporal transcoding is a process that skips some frames in order to change the frame rate of the video sequence and to allocate more bits to remaining frames without decreasing the video quality for not skipped frames. In addition, frame-rate conversion is also needed when the end system supports only a lower frame-rate. In temporal transcoding, the main issues are:

- to recompute the motion vectors not still valid, because they point to discarded frames;
- to recompute the prediction errors according to the new motion vectors;
- to choose the frames to be skipped (*frame skipping policy*).

This technique is the main one for our work, and will be described in greater detail in the next sections.

Chapter 4: Temporal video transcoding features

In video transcoding, the compressed video bitstream is often converted to a reduced frame-rate video bitstream in order to decrease the bit rate. One way to perform this is to decompress the compressed video bitstream, drop several specific frames and compress the non skipped frames to another compressed video bitstream. This method has high computational complexity, since when some frames are skipped, the incoming motion vectors are no more valid because they point to dropped frames and a motion estimation process is needed in order to compute the new motion vectors of the non-skipped frames. These new motion vectors can be found by the "Motion Vector Composition" algorithms. After that, a "Refined Motion Estimation" can be applied to the composed motion vectors. These techniques, illustrated in Section 4.1, permit to reduce the computational complexity of the motion estimation process and to achieve a good video quality of the transcoded frames. According to the new motion vectors, computing the new prediction errors of the non skipped frames is needed, as showed in Section 4.2.

4.1 Motion Vector Composition

A situation where one frame is dropped is illustrated in Figure 12. Since frame R_{t-1} is dropped, we need to find a motion vector for MB_t macroblock pointing to the best prediction for MB_t in frame R_{t-2}. One possible way to generate such a motion vector without performing motion estimation is to add vector (u_t, v_t) to vector (u_{t-1}, v_{t-1}) in order to compose (u_t^s, v_t^s).

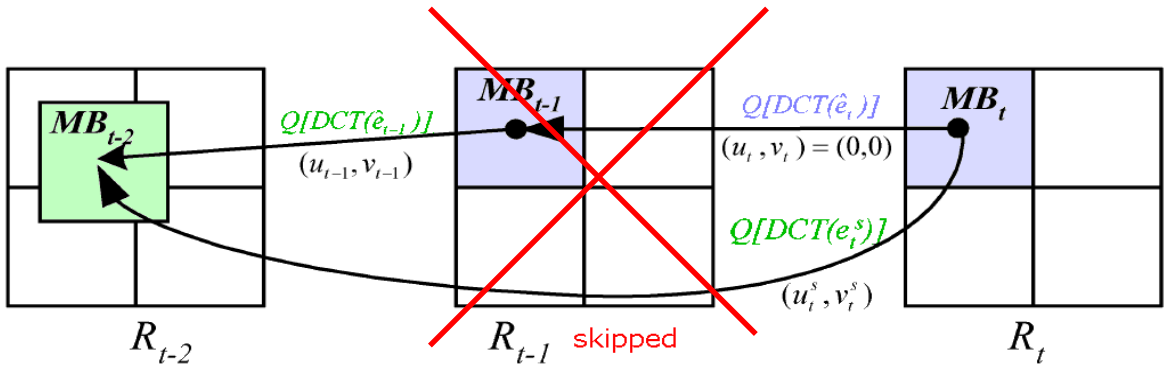


Figure 12. Motion Vector Composition: case motion vector null

In case the motion vector (u_t, v_t) is equal to (0, 0) (Figure 12), it points to macroblock MB_{t-1}, so the vectors (u_t, v_t) and (u_{t-1}, v_{t-1}) are both available in the incoming bitstream and the MB_t

macroblock is named macroblock non-Motion Compensated (MC). In general MB_{t-1} is not a macroblock, but a reference area of 16*16 pixels that overlaps four macroblocks in the skipped frames (Figure 13), so (u_{t-1}, v_{t-1}) is not available in the incoming bitstream.

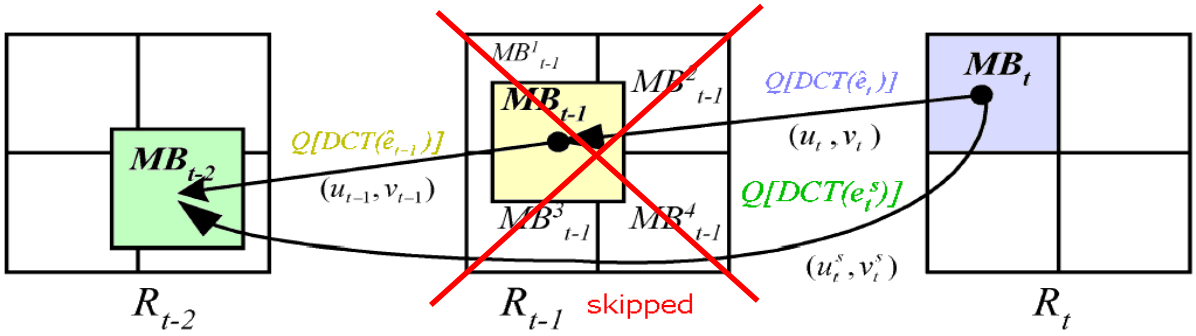


Figure 13. Motion Vector Composition: case motion vector non null

It is possible to use four algorithms named Bilinear Interpolation (BI), Forward Dominant Vector Selection (FDVS), Telescopic Vector Composition (TVC), and Activity Dominant Vector Selection (ADVS) for computing an approximation of (u_{t-1}, v_{t-1}). If the frame R_{t-2} also is skipped, the vector (u_t^s, v_t^s) is no more valid. By the Motion Vector Composition, it is possible to find a new motion vector (u_{t-2}, v_{t-2}) for MB_{t-2} reference area. The vector (u_{t-2}, v_{t-2}) is added to (u_t^s, v_t^s) in order to obtain a new motion vector for MB_t pointing to the last transcoded frame. In general, if k consecutive frames from $t-k$ to $t-1$ are dropped during transcoding, the motion vector of the macroblock of the current frame can be composed repeatedly applying the Motion Vector Composition. The resultant motion vector will be

$$(x,y)_t = \left(\Sigma_{d=k}^{1}(V_x)_{t-d} + (I_x)_t, \Sigma_{d=k}^{1}(V_y)_{t-d} + (I_y)_t\right) \quad (4.1)$$

where $(V_x, V_y)_{t-d}$ is the motion vector selected by the Motion Vector Composition at the frame $(n-d)$, and $(I_x, I_y)_t$ is the incoming motion vector of the frame (t).

The motion vector obtained by the Motion Vector Composition is an approximated value of the optimal motion vector. The application of a motion estimation is needed to obtain the refined motion vector. The full search motion estimation can be applied to exhaustively compute all checking points in a new small search window, but this has a high computational complexity. Several schemes for motion estimation in transcoders are discussed in [30],

where it is showed that by using refined motion estimation (RME) in a much reduced search area, it is possible to achieve near optimal outgoing motion vectors with a quality close to the full scale motion estimation. In [16][18], two refined motion estimation schemes are proposed that reduce the computational complexity of the full motion estimation and achieve similar performance results to the fast motion-estimation algorithm in term of PSNR.

4.1.1 Bilinear Interpolation

The bilinear interpolation is defined as [15]:

$$MV_{int} = (1-\alpha)(1-\beta)\ MV_1^{n-1} + (\alpha)(1-\beta)\ MV_2^{n-1} + (1-\alpha)(\beta)\ MV_3^{n-1} + (\alpha)(\beta)\ MV_4^{n-1} \quad (4.2)$$

where MV_1^{n-1}, MV_2^{n-1}, MV_3^{n-1} and MV_4^{n-1} are the motion vectors of the four macroblocks that overlap the reference area in the skipped frame pointed by the incoming motion vector, α and β are determined by the horizontal and vertical pixel distance of this reference area from the MV_1^{n-1} (Figure 14). The selected motion vector is MV_{int}.

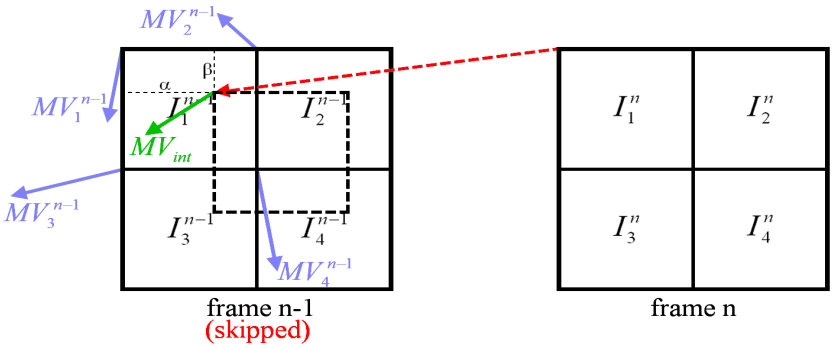


Figure 14. Bilinear Interpolation

4.1.2 Forward Dominant Vector Selection

In [16], the Forward Dominant Vector Selection algorithm is proposed. This algorithm selects one dominant motion vector among the vectors of the four macroblocks that overlap the reference area in the skipped frame. This dominant vector (MV_{fdvs}) is defined as the motion vector of the dominant macroblock. The dominant macroblock is a macroblock that has the largest overlapping area with the reference area pointed by the incoming motion vector.

For example, $MV_{fdvs} = MV_2^{n-1}$ in Figure 15.

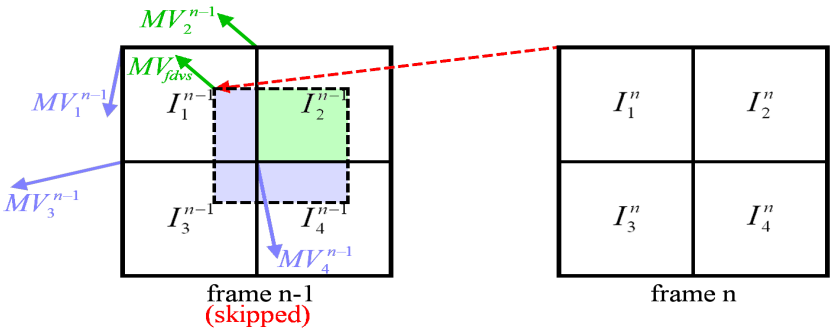


Figure 15. Forward Dominant Vector Selection

This algorithm presents a lower computational complexity than the bilinear interpolation. The approximation of MV_{fdvs} is more accurate when the overlapping area of the dominant macroblock with the reference area is larger. However, when the overlapping areas among the four near macroblocks are very close, the motion vector decided by FDVS may not be meaningful.

In [35], the conventional FDVS method is improved to reflect the effect of the macroblock types in the skipped frames.

In [29], the Bi-direction Dominant Vector Selection (BDVS) is presented. It is based on FDVS algorithm but it is designed to re-estimate the dominant motion vectors in popular I-B-P frame structure video sequences that are not considered in FDVS.

4.1.3 Telescopic Vector Composition

A simple algorithm is Telescopic Vector Composition [17], that selects, in the skipped frame, the motion vector (*MVtvc*) of the macroblock corresponding to the macroblock in the current frame. For example, $MVtvc = MV_1^{n-1}$ in Figure 16.

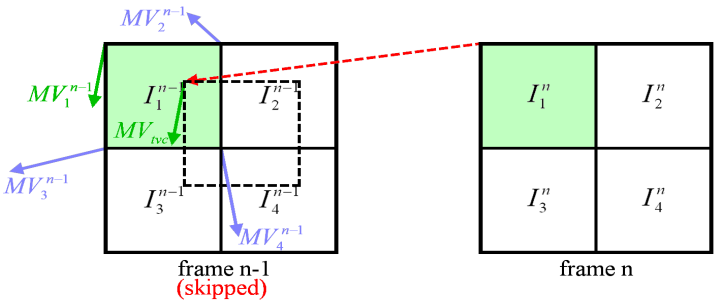


Figure 16. Telescopic Vector Composition

The basic idea is that in videos with small motion, the motion vectors are very small, so the reference area pointed by the incoming motion vector will always overlap most with the corresponding macrobloclock in the skipped frame. For this reason, the results obtained by TVC and FDVS can be very close.

4.1.4 Activity Dominant Vector Selection

The Activity Dominant Vector Selection algorithm presented in [18] is based on the activity of the macroblocks for the choice of the motion vector. The activity of a macroblock is represented by the number of nonzero quantized DCT coefficients (*NZ*) of the prediction errors of the blocks that belong to that macroblock. The ADVS algorithm selects the motion vector (*MVadvs*) of the macroblock with the biggest activity among those that overlap the reference area pointed by the incoming motion vector. Other statistics can also be used, such as the sum of the absolute values of DCT coefficients. For the case shown in Figure 17, FDVS chooses the motion vector of I_4^{n-1} macroblock as the dominant vector ($MVadvs = MV_4^{n-1}$), since $NZ(I_4^{n-1})$ is larger than $NZ(I_2^{n-1})$, although $NZ(I_4^{n-1})$ only covers two blocks, which are smaller than the four blocks covered by $NZ(I_2^{n-1})$.

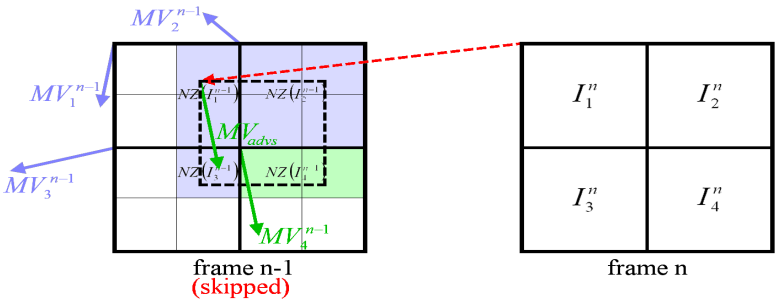


Figure 17. Activity Dominant Vector Selection

The idea of this algorithm is to select the motion vector of the macroblock with maximum activity (*NZ*) that corresponds to larger prediction errors. The bigger is the activity of the macroblock, the more significant is the motion of the macroblock. Since the quantized DCT coefficients of prediction errors are available in the incoming bitstream of transcoder, the computation for counting the nonzero coefficients is very low.

4.2 New prediction errors computation

In temporal transcoding, after skipping some frames, according to the new motion vectors, it is needed to compute also the new prediction errors. There are two architectures in literature that perform this and they are presented in the next sections. The skipped frames must be decompressed completely, since are used as reference frames for the decoding of successive non-skipped frames.

4.2.1 Prediction errors in Pixel Domain

In pixel-domain transcoder architectures, a frame R_t is decoded with R_{t-1} frame reference as:

$$R_t(\mathrm{i,j}) = R_{t-1}(\mathrm{i}+u_t,\ \mathrm{j}+v_t) + e_t(\mathrm{i,j}) + \Delta_t(\mathrm{i,j}) \quad (4.3)$$

where e_t represents the prediction errors (residuals) between the current frame R_t and the motion-compensated frame R_{t-1}, Δ_t represents the reconstruction errors due to quantization in the remote encoder and (u_t, v_t) represents the motion vectors of the current frame related to the reference frame. If R_{t-1} is dropped, by Motion Vector Composition and Refined Motion Estimation it is possible to find a new motion vector $(u_t^s\ v_t^s)$ for R_t that points to previous non skipped frame R_{t-2}. Regard to this new motion vector, the new prediction errors $e_t^s(\mathrm{i,j})$ are computed and coded by DCT and quantization. The remote decoder reconstructs the frame R_t as:

$$R_t^s(\mathrm{i,j}) = R_{t-2}^{\ s}(\mathrm{i}+u_t^s,\ \mathrm{j}+v_t^s) + e_t^s(\mathrm{i,j}) + \Delta_t^s(\mathrm{i,j}) \quad (4.4)$$

where Δ_t^s represents the re-quantization errors due to re-encoding in the transcoder. These re-encoding errors affect the video quality of the non skipped frames, with a degradation of the PSNR of about 3.5 dB on average, compared to that of the same pictures directly decoded without the transcoding process [19].

4.2.2 Prediction errors in DCT Domain

In order to reduce the complexity of the transcoding process, and to avoid re-encoding errors Fung proposed in [31][32] a new frame skipping transcoder architecture based on the direct addition of DCT coefficients of the prediction errors.

We show in Figure 18, a situation where the frame R_{t-1} is dropped, and the macroblock MB_t is non-Motion Compensated. We assume that MB_{t-1} represents the best matching macroblock for the current macroblock MB_t, $Q[DCT(\hat{e}_t)]$ and $Q[DCT(\hat{e}_{t-1})]$ are the prediction errors of MB_t and MB_{t-1} respectively.

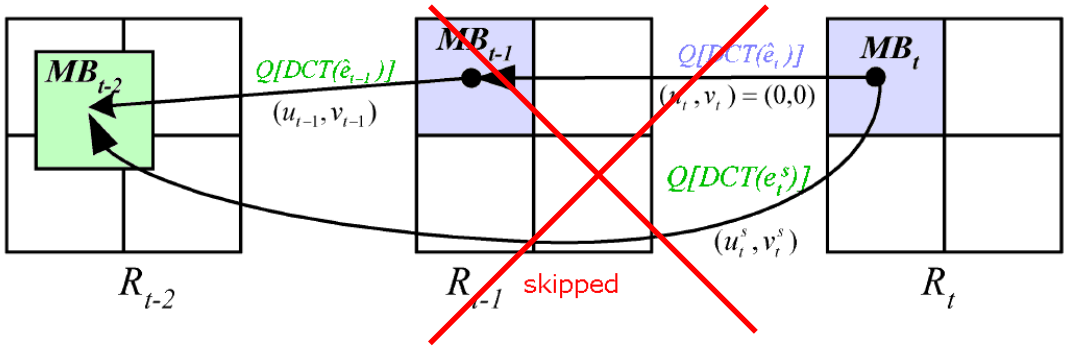


Figure 18 New prediction errors computation for non-MC macroblocks

In this proposed architecture, the new quantized DCT coefficients of prediction errors $Q[DCT(\hat{e}_t^s)]$ are computed directly in DCT domain and are given by:

$$Q[DCT(\hat{e}_t^s)] = Q[DCT(\hat{e}_t)] + Q[DCT(\hat{e}_{t-1})] \quad (4.5)$$

Since in this case MB_t is non-Motion Compensated, the quantized DCT coefficients $Q[DCT(\hat{e}_t)]$ and $Q[DCT(\hat{e}_{t-1})]$ are available in the input bitstream to the transcoder. The transcoding complexity is reduced since it is not necessary to perform motion compensation, DCT, quantization, inverse DCT and inverse quantization. Furthermore, since requantization is not necessary for non-Motion Compensated macroblocks, re-encoding errors Δ_t^s mentioned in (4.4) are also avoided.

Many real-world image sequences have a smooth motion that varies slowly, so over 70% of the macroblocks are non-Motion Compensated. By using a direct addition of the DCT coefficients in the frame-skipping transcoder, the sequences containing more non-Motion Compensated macroblocks can reduce the computational complexity and the re-encoding errors more significantly.

In Figure 19, we show the case of a Motion Compensated macroblock MB_t. In this case, direct addition of the prediction errors cannot be employed since MB_{t-1} is not a macroblock and so $Q[DCT(\hat{e}_{t-1})]$ is not available from the incoming bitstream.

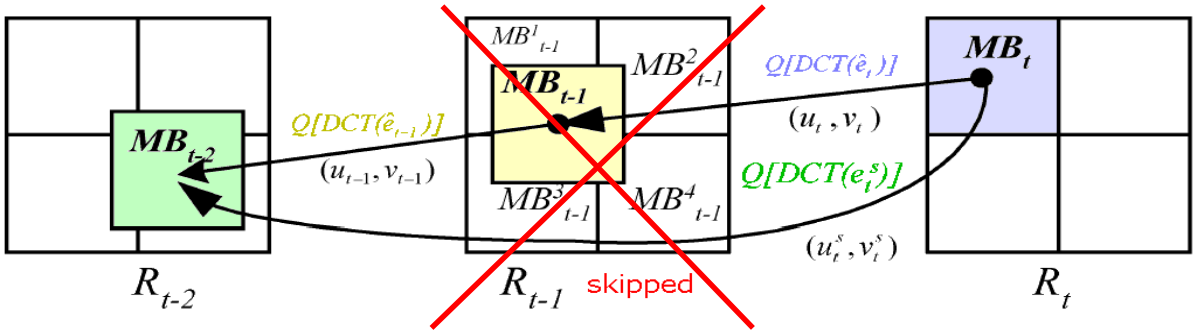


Figure 19 New prediction errors computation for MC macroblocks

It is possible to use the incoming quantized DCT coefficients of the macroblocks MB_{t-1}^{1}, MB_{t-1}^{2}, MB_{t-1}^{3}, MB_{t-1}^{4} that overlap MB_{t-1} for computing $\hat{e}_{t-1}$ First, inverse quantization and inverse DCT of coefficients of the blocks that overlap MB_{t-1} are performed, to obtain their corresponding prediction errors in the pixel-domain. These prediction errors are added to the motion compensated segments of the previous non skipped frame to obtain all pixel in MB_{t-1}. It is possible to use the motion vector composition to compute the motion vector (u_{t-1}, v_{t-1}) of MB_{t-1} pointing to a reference area in R_{t-2}. The prediction errors $\hat{e}_{t-1}$ are obtained by subtracting MB_{t-1} from the corresponding motion compensated reference area MB_{t-2} in the previous not skipped frame. DCT and quantization are applied to $\hat{e}_{t-1}$ to obtain $Q[DCT(\hat{e}_{t-1})]$. The new quantized DCT coefficients $Q[DCT(\hat{e}_t^s)]$ of a Motion Compensated macroblock can then be computed by adding $Q[DCT(\hat{e}_{t-1})]$ to the incoming $Q[DCT(\hat{e}_t)]$. The requantization introduced for computing $Q[DCT(\hat{e}_{t-1})]$ brings additional re-encoding errors Δ_{t-1}^{s}.

These errors degrade the quality of the reconstructed frame. Since each non skipped P-frame is used as a reference frame for the following non skipped P-frame, quality degradation propagates to later frames in a cumulative manner. Figure 20 shows how re-encoding errors can lead to accumulated errors. In this figure, Δ_2^s is introduced for the quantization of $Q[DCT(\hat{e}_2)]$ and such errors have the effect of degrading the quality of the reconstructed area MB_3. When pixels in MB_3 are used as reference for the next non skipped frame (for example R_5^s), Δ_2^s affects the formation of $Q[DCT(\hat{e}4)]$ and this error is accumulated in the reconstruction of MB_5. These accumulated errors become significant in the sequence containing a large amount of Motion Compensated macroblocks.

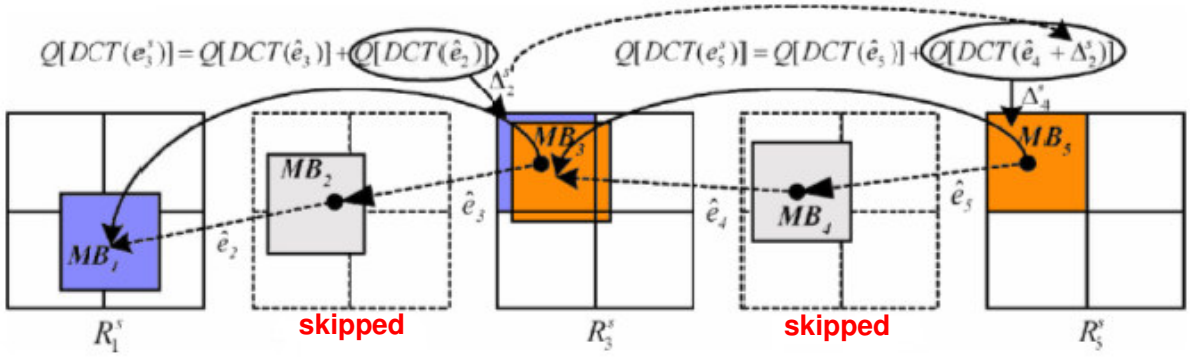


Figure 20. Effect of re-encoding error with error compensation

A technique introduced to minimizing the visual degradation caused by this phenomenon is the error compensation. The re-encoding errors are computed by:

$$\Delta_{t-1}^{s} = \mathrm{DCT}^{-1}(\mathrm{DCT}(\hat{e}_{t-1}) / q * q) - \hat{e}_{t-1} \quad (4.6)$$

where q is the quantization parameter.

These re-encoding errors are stored and added to the prediction errors of Motion Compensated macroblocks in the successive P frames. For example as show in Figure 20, during the computation of MB_3, re-encoding error Δ_2^s is stored. During the computation of MB_5, Δ_2^s is added to $\hat{e}_4$ computed by MB_3 which is affected by the re-encoding error Δ_2^s.

This technique cannot entirely avoid the propagation of re-encoding errors, but it reduces their effect on the visual quality of the transcoded frames.

Chapter 5: Main research results

In this section we present our main research results.

5.1 Temporal transcoding in MPEG4

The first coding standard that we studied has been MPEG4. We chosen the MoMuSys-FPDAM1-1.0-021202_nctu version of this coding standard [42] and we implemented two different temporal transcoding architectures that we show below.

5.1.1 DFS and FSC architectures

The first temporal transcoding architecture that we developed is called DFS (Dynamic Frame Skipping). This architecture 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 this architecture, the motion vectors are computed by one of the four MVC algorithms, and RME procedure previously mentioned in Section 4.1. The prediction errors are computed in the pixel domain. This architecture is shown in Figure 21. The behaviour 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 behaviours is represented in Figure 21 by the switch "PS/PT". In addition, also transcoding or skipping of the current frame determines a different behaviour (switch "CS/CT"). The left part of Figure 21 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.

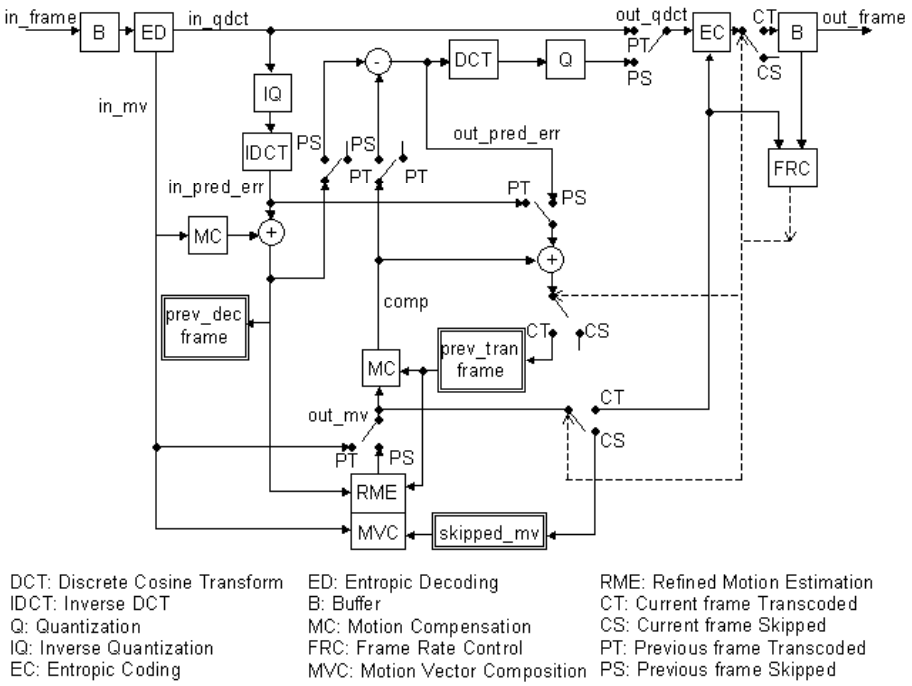


Figure 21. Temporal transcoder architecture

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 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*. Reconstructed frames are then skipped or placed in the buffer for being transmitted. An important issue in temporal transcoding is the choice of frames to be skipped. The frame rate control developed in DFS architecture, dynamically adjusts the number of skipped frames according to the motion activity. The motion activity gives a measure of the motion in a frame and frames with a lot of motion are not skipped. We are going to better explain the motion activity measure below. A different temporal transcoding architecture, called Frame Skipping Control (FSC) consists in computing the prediction errors

in the DCT domain as presented in Section 4.2.2. This way of computing prediction errors produces re-encoding errors, then in FSC architecture the frames are skipped taking into account re-encoding errors effect and the motion activity.

5.1.2 Motion based frame skipping policy

In [15], a frame rate control scheme, which can dynamically adjust the number of skipped frames according to the motion activity, is presented. The motion activity of the current frame t, (MA_t), is the sum of the motion activities of all the macroblocks $(MA)_m$ of the frame, and is defined as:

$$MA_t = \sum_m (MA)_m \quad (5.1)$$

where $(MA)_m$ is the sum of the vertical and horizontal components of the motion vector of macroblock m, namely:

$$(MA)_m = (|x_i| + |y_i|) \quad (5.2)$$

This motion activity is compared with a dynamic threshold value, computed according to the motion activity of the previous frames and the number of transcoded frames. When a frame is skipped, the remote decoder replaces the missing frame with the previous transcoded frame. The basic idea of this scheme is that if the motion activity is larger than the threshold, the frame can not be skipped since it has considerable motion, and it is not possible to have a good approximation of this frame by using the previous transcoded frame only. We implemented this motion-based skipping policy in DFS architecture. Simulation results show that by using this approach presented above the video displayed at the receiver site is smoother.

5.1.3 Motion activity and Re-encoding errors in frame skipping

When more frames are dropped, re-encoding errors in motion compensated macroblocks cannot be avoided entirely, even if error compensation schemes are applied (as mentioned in the Section 4.2.2). In [19], a frame skipping strategy which takes in account the effect of the re-encoding errors is proposed. The goal of this strategy is to minimize the re-encoding errors

as well as to preserve motion smoothness. A frame skipping metric based on the motion activity and re-encoding errors is defined as:

$$FSC_t(MA_t, RE_t) = \frac{\sum_{m=1}^{M} (MA_t)_m}{\sum_{m=1}^{M} (RE_t)_m} \qquad (5.3)$$

where M is the total number of macroblocks in the current frame t, $(MA_t)_m$ is the motion activity of the m_{th} macroblock defined in (5.2), and $(RE_t)_m$ are the re-encoding errors of the m_{th} macroblock defined as:

$$(RE_t)_m = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} \Delta_{t-1}^{s}(i, j) \qquad (5.4)$$

where N is the size of macroblocks of the current frame and Δ_{t-1}^{s} are defined by (4.6). If the value of $FSC_t(MA_t, RE_t)$ for a not skipped frame exceeds a predefined threshold $T_{FSC,}$ the frame is not skipped since it has considerable motion, and the previous not skipped frame is not sufficient to represent the current frame. A large value of $\sum_{m=1}^{M} (RE_t)_m$ implies more re-encoding errors, and reduces the value of $FSC_t(MA_t, RE_t)$. So, if this value is smaller than the threshold, the frame can be skipped since it contains many re-encoding errors. The threshold $T_{FSC,}$ is set initially to a value T_{init}, and can be dynamically updated with a granularity of T_{step} in order to stabilize the outgoing frame rate f_o according to the target frame rate f_T of the transcoder in this way:

- if $f_o > f_T$, increase T_{FSC} by T_{step};
- if $f_o < f_T$, decrease T_{FSC} by T_{step};
- otherwise, keep the current value of T_{FSC}.

We implemented this strategy in FSC architecture. Simulation results show that it minimizes the re-encoding errors and preserves the motion smoothness of the transcoded frames.

5.1.4 Comparison between temporal and quality transcoding

We implemented a simple 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. We compared this transcoder with our temporal transcoder realized with the DFS and FSC architectures, over several benchmark videos. The results, in terms of PSNR (a measure indicating the quality of the transcoded sequence) show that a better performance is achieved by quality transcoder (QT) for videos with a lot of motion and by temporal transcoderng (DFS and FSC) for videos with little motion. Moreover, we observed that DFS architecture achieves a better performance than FSC one, since in the latter, many frames are skipped because of re-encoding errors. We had similar results considering different MVC algorithms (Bilinear Interpolation (BI), Telescopic Vector Composition (TVC), Forward Dominant Vector Selection (FDVS), Activity Dominant Vector Selection (ADVS)). These results are presented in Figure 22.

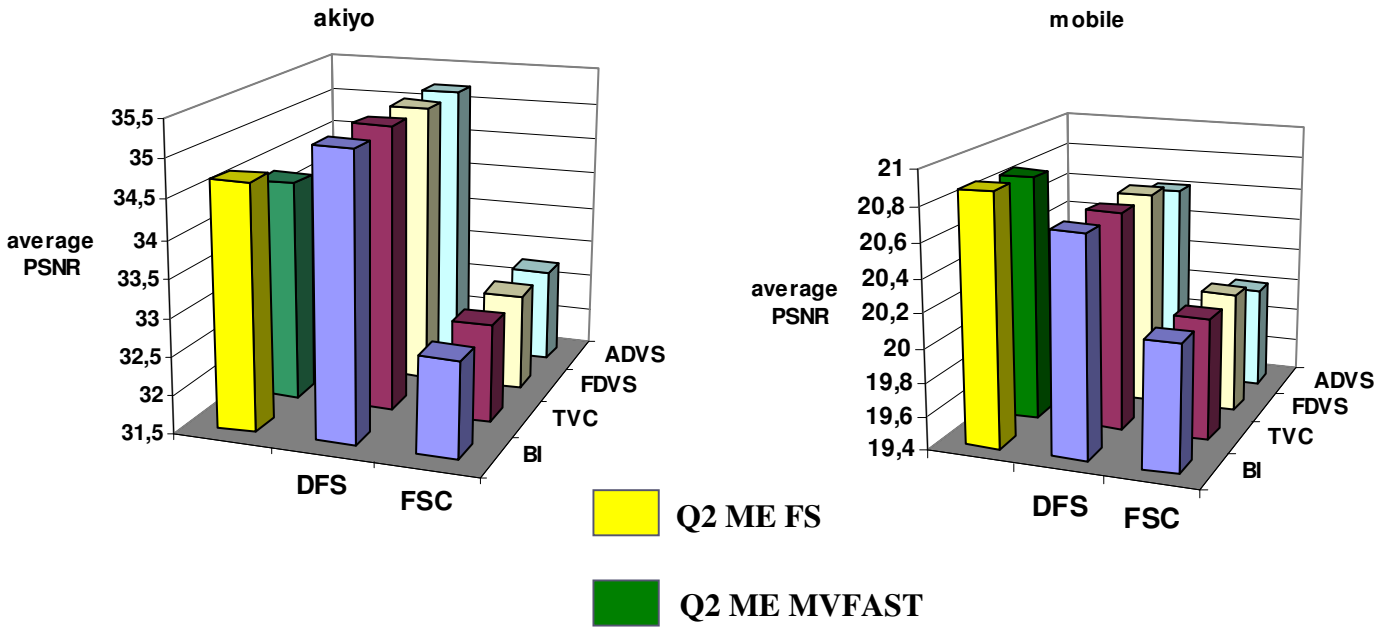


Figure 22. MPEG4 transcoding architectures evaluation

5.1.5 Buffer based frame skipping policy

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<= $B_{lower}S$)&(L+L(f)<=S) transcode f
else
if ((L>= B_{upper} S) skip f
else
if(L + L(f)>=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, in order to improve the quality of the transcoded video sequence.

5.1.6 Random based frame skipping policy

Randomization is used for studying the behaviour 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. 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 >=L) transcode f
else skip f.

5.1.7 Weighted motion activity in frame skipping

In Section 5.1.3, we reported a motion based frame skipping policy proposed in literature that we have implemented and tested. 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 5.2, and proposed the following one:

$$MA = \sum_{m} k^{|x^m|} + k^{|y^m|} \qquad (5.5)$$

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)<= 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.

5.1.8 Consecutive frame 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 Γ=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/ Γ) frames and skip N(1-1/ Γ) frames. Every Γ successive frames, one of them should be transcoded, and Γ-1 should be skipped for distributing uniformly the skipped frames. This strategy forces the transcoder to skip Γ-1 consecutive frames, in order to prevent the number of consecutive skipped frames to become larger than Γ-1. We show below the pseudo-code of the whole strategy.

MaxConsecutiveSkipping Policy*(frame f):*
if (numConsecutiveSkippedFrames < Γ)
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.

In table 1 we show the performance of our frame skipping policies. We considered two metrics: the number of transcoded frames (indicating the video sequence smoothness), and the PSNR. 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. 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. We report in Figures 23 and 24 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. 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 24, 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.

Table 1. Frame skipping policies evaluation

	mobile			foreman			coastguard		
	Frames	PSNR1	PSNR2	Frames	PSNR1	PSNR2	Frames	PSNR1	PSNR2
Standard transcoding conditions (IR=128, R=64 kbps)									
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 conditions(IR=128, R=32 kbps)									
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

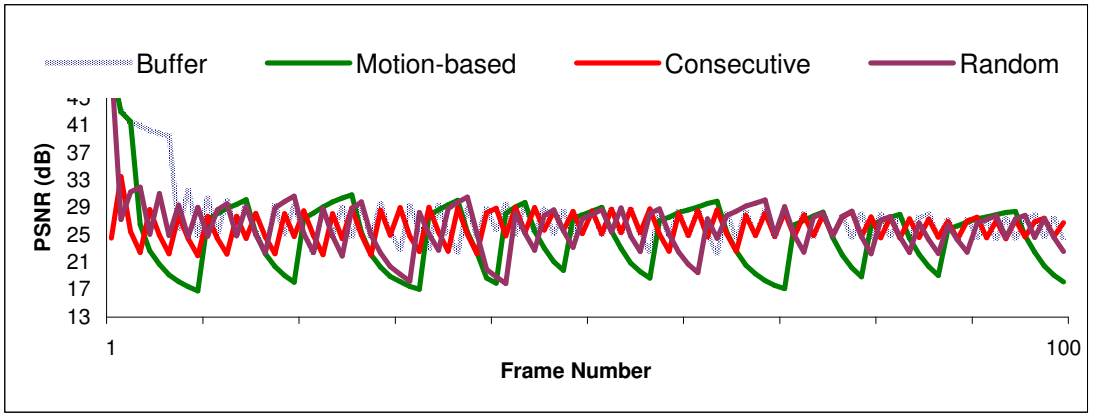


Figure 23. Mobile video sequence IR=128, R=64 kbps

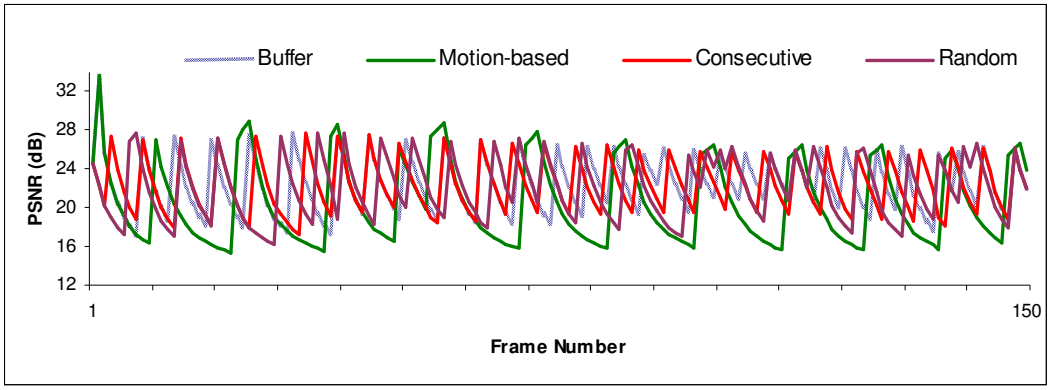


Figure 24. Mobile video sequence IR=128, R=32 kbps

Publications

M.A. Bonuccelli, F. Lonetti, F. Martelli, "***Temporal Transcoding for Mobile Video Communication***", Proc. 2th International Conference on Mobile and Ubiquitous System: Networking and Services, MobiQuitous 2005, San Diego, CA, USA, July 17-21, 2005.

M.A. Bonuccelli, F. Lonetti, F. Martelli, "***Video Transcoding Architectures for Multimedia Real Time Services***", ERCIM News No. 62, July 2005, p. 39-40.

5.2 Temporal transcoding in H.263

We studied the H.263 coding standard [43] in accordance with [44], and we implemented a temporal transcoder with DFS architecture explained in Section 5.1.1 that had showed the better performance results compared with FSC architecture. We tested this temporal transcoder with buffer based frame skipping policy, motion-activity based frame skipping policy and consecutive skipping policy. Moreover we focused on time constraints of our temporal transcoding in order to apply it in a real-time context. We proposed a new frame skipping policy that we called *size-prediction policy*. It greatly reduces the computation time of transcoding process and it is presented below.

5.2.1 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:

$$l(f) = \alpha \ln(f+1) \qquad (5.6)$$

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 (5.6), 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 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 Section 5.1.5 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. The performance of this policy is compared to that of the buffer-based one. The main performance results are presented in table 2. We considered three metrics: number of transcoded frames (indicating the video sequence smoothness), PSNR 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. 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%). The pseudo-code of size-prediction policy is shown below.

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/ρ
while ((L > B_{lower}(S)) & (L + l(f) ≥ S))
transcode f

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

	Buffer-based				Size prediction			
	frames	PSNR1	PSNR2	Time	frames	PSNR1	PSNR2	Time
IR=256, 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
forema	110	32.38	44.49	9.4	110	32.23	44.29	6.5
coastgu	118	32.39	41.69	9.5	113	31.82	41.57	6.1
IR=64, 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
forema	112	29.88	36.10	7.7	109	29.29	36.45	4.6
coastgua	144	31.36	34.82	7.1	129	30.60	34.93	4.4

We implemented also a H.263 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). As shown in Table 3, our temporal transcoder has a comparable computation time than the quality one, but obviously skips more frames, so it produces a sequence with lower 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 3, 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 degradation of quality transcoder.

Table 3. Temporal vs quality H.263 transcoder

	Temporal transcoder			Quality transcoder		
	frames	PSNR1	Time(sec)	frames	PSNR1	Time(sec)
IR=256, 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	32.77	9.7
coastguard	118	32.39	9.5	300	32.47	9.3
IR=64, 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

5.2.2 Rate control algorithms

In our simulations, we observed that in temporal transcoding the quality of transcoded frames is also influenced by the rate control algorithm of the front encoder, especially at low bit-rates, where it is more evident the quality degradation introduced by a non efficient rate control algorithm. The rate control scheme is not defined in the standard, thus different strategies can be implemented in each encoder design. Our objective was to develop a new rate control technique able to give stable quality, with reasonable computation complexity for practical applications. Many rate control schemes have been proposed in literature. In general, they operate at frame layer or macroblock layer. A frame-layer rate control assigns a target number of bits to each video frame and, at a given frame, the block-layer rate control selects the block quantization parameters to achieve the assigned target. Some frame-layer rate control approaches use simple formulas, but these simple methods generally do not achieve the target number of bits accurately. Other approaches use various rate-distortion strategies to assign a target number of bits to each frame [45]. However, since they usually use either an iteration method for optimal bit allocation or a pre-analysis method on a group of frames before encoding, they produce time delay or high computational complexity. The TMN8 rate control for the H.263 standard, uses a frame-layer rate control to select a target number of bits

for the current frame, and a macroblock layer rate control to select the values of the quantization step sizes for the macroblocks. We evaluated TMN5 and TMN8 rate control algorithms proposed in encoder standard [46]. We implemented two new approaches for the rate control proposed in literature: the ρ domain, based on the number of quantized coefficients called ρ [47], and the Perceptual rate control, based on the different perception of different parts of the image by the human eyesight [48]. Moreover, we proposed a new rate control scheme that we called Multiple zone (Activity) operating at frame and macroblock layers. At frame layer, it determinates a bit-budget called S_i ,considering a sliding window of 5 frames. Initially, the bit-budget is $S_i = N_w$(R/F), where N_w is the size of the sliding window, R and F are the bit and frame rates respectively. The bit-budget for the current frame is

$$R_i^t = S_i \times \frac{SAD(f_{ref}, f_{curr})}{Sum\_SAD_i} \quad (5.7)$$

where f_{ref} and f_{curr} are respectively the reference frame and the current frame, Sum_SAD_i is the sum of computed SAD until instant i in the sliding window. In order to prevent buffer underflow and overflow we set the bit-budget of the current frame as

$$\tilde{R}_i^t = \max\left\{\min\left\{R_i^t, \alpha \times (R/F) + (R/F) - W_{prev}\right\}, \beta \times (R/F) + (R/F) - W_{prev}\right\} \quad (5.8)$$

where α and β are respectively equal to 0.9 and 0.05.
In the sliding window we have that

$$S_{i+1} = S_i - \tilde{R}_i^t + R/F \quad (5.9)$$

where Sum_SAD_i is updated after N_w frames.

At macroblock layer the frame is divided in multiple zones, each one composed by a set of macroblocks. The goal is to encode with greater quality and then with lower quantization parameters, the central zone that most attracts the human eyesight. This is possible by properly tuning the quantization parameters as in [48]. Another version of this algorithm assumes lower quantization parameters for macroblocks in zones of the frame that present a greater motion activity compared with the total motion activity of the frame. By simulation we can see that our algorithm achieves a good performance mostly in video sequence with little motion and low bit-rate. By our tests we observed also that for each type of video

sequence a good approach is that one based on ρ-domain while a lower performance is achieved by the perceptual algorithms.

Publications

M.A. Bonuccelli, F. Lonetti, F. Martelli, "***A fast skipping policy for H.263 video transcoder***", Proc. 12th International Workshop on Systems, Signals & Image Processing, IWSSIP '05, Chalkida, Greece, September 22-24, 2005.

5.3 Temporal transcoding in H.264

H.264/AVC is the newest international video coding standard [36]. In 1998, the Video Coding Experts Group (VCEG) issued a call for proposal on a project called H.26L, with the target to double the coding efficiency. In 2001, VCEG and MPEG formed a Joint Video Team (JVT). In March 2003, the first draft of H.264 standard was released. The main characteristic of this standard with respect to previous standards is a greater bit-rate reduction to the detriment of an higher complexity. The new standard is designed for technical solutions including different application areas: wire-line and wireless real-time conversational services, streaming media, video-on-demand services over ISDN, LAN, wireless networks, multimedia messaging services (MMS). To address this need for flexibility and customizability, the H.264/AVC design covers a Video Coding Layer (VCL), which is designed to efficiently represent the video content, and a Network Abstraction Layer (NAL), which formats the VCL representation of the video and provides header information in a manner appropriate for conveyance by a variety of transport layers or storage media. Relative to prior video coding methods, there are some features of the design that enable enhanced coding efficiency [51]: variable block-size motion compensation with small block sizes (4×4), quarter-sample-accurate motion compensation, motion vectors over picture boundaries, multiple reference picture motion compensation, deblocking filtering, arithmetic entropy coding (CABAC) and context-adaptive entropy coding (CAVLC). Our goal was to implement a temporal transcoder by using H.264 reference software. We chose the JM Reference Software version 9.7 [49], that was the most complete, in accordance to [50]. The main problem with H.264 reference software has been that it was very slow. By profiling the H.264 encoder, we realized the large times spent in each encoding function. As expected, the encoding time is greatly dominated in partitioning the macroblocks. We operated some modifications to the reference software in order to obtain acceptable encoding times that we explain in the next section.

5.3.1 Coding standard optimization

In order to optimize the reference software, we first operated purely software modifications. In particular, instead of computing all half and quarter pixels in two rounds, we compute them in only one round. Then, we made more important changes about the computation of partitioning of the coded frame. Instead of using the SAD (Sum of Absolute Differences) measure as decision parameter for the optimal partitioning of the coded frame, we used other metrics: the number of differences in terms of pixels, the maximum difference value, the average difference value. These metrics are compared with proper self-adjusting thresholds. These metrics resulted faster in determining the optimal frame partitioning. By simulations we observed that both our optimizations gave an important reduction of encoding time with a loss of few dB in PSNR.

After optimizing the reference software, we developed a temporal transcoder, we adopted the same architecture used for MPEG4/H.263 temporal transcoders explained in Section 5.1.1. About the frame partitioning, we assumed that the transcoder keeps the same partitions of the remote encoder: this is the most efficient solution in term of computation time. Due to variable macroblock partition (16 motion vectors for each macroblock), the motion vector composition was not trivial, and we adapted the MVC algorithms present in literature to be used in variable partitioning, as shown in Section 5.3.2. We present a new motion vector composition algorithm for H.264 transcoder in Section 5.3.3.

5.3.2 Multi-level Motion Vector Composition

In order to consider the variable partition of H.264 reference frame, we adopted a new scheme of motion vector composition, operating at multiple levels. The goal is to find a motion vector in the last skipped frame, to be composed with the motion vector of the current frame, in order to obtain a motion vector for the current frame that points to the last skipped frame as we illustrated in Section 4.1. The problem in H.264 is that, for each macroblock of the skipped frame, there are many motion vectors corresponding to different partitions of such macroblock. We adopted a multi-level scheme illustrated in Figure 25. At first level, we consider the reference frame, and the macroblocks overlapping the reference area pointed by the current motion vector. For each of these macroblocks, we go down at lower levels until all partitions and sub-partitions overlapping the reference area are considered. At each level, beginning from the lowest one, and for each partition overlapping the reference area, we choose a motion vector. This motion vector is that one of the partition overlapping the

reference area or a composition of the motion vectors of sub-partition overlapping the reference area. The composition is performed according to one of the motion vector composition algorithms presented in Section 4.1. At the end, we obtain four motion vectors at level 0, that are composed according to one of the motion vector composition algorithms presented in Section 4.1. The result of this composition is added to the current motion vector in order to obtain the new motion vector pointing to the last non skipped frame. We implemented this multi-level motion vector composition scheme by considering the Bilinear Interpolation (BI) and the Telescopic Vector Composition (TVC) algorithms presented respectively in Section 4.1.1 and 4.1.3.

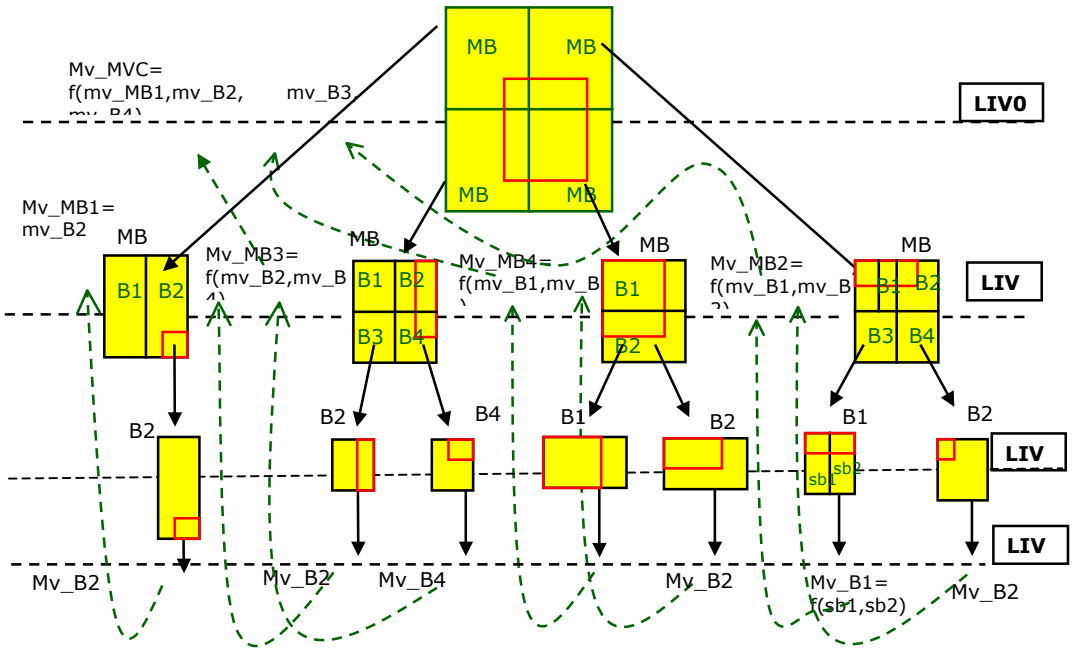


Figure 25. Multi-level motion vector composition scheme

5.3.3 New Motion Vector Composition Algorithm

We proposed a new motion vector composition algorithm for H.264. The basic idea of this algorithm is looking for a reference area, the most similar one to the macroblock of the current motion vector that is no more valid. For each partition overlapping the reference area, we consider its motion vector and we consider, in the previous non-skipped frame, an area

pointed by this motion vector with a size equal to that of the current macroblock. We compute the differences between this area and the current macroblock and we choose the area that minimizes this difference according to the law:

$$mv_f = \arg\min_{i \in S} MSE(A, A_i) = \arg\min_{i \in S} 1/M \times N |A - A_i|^2 \quad (5.9)$$

where A is the current macrblock, A_i is the reference area pointed by the motion vector mv_i, S is the set of macroblock partitions overlapping the reference area, and mv_f is the chosen motion vector. We can see the behaviour of this algorithm in Figure 26.

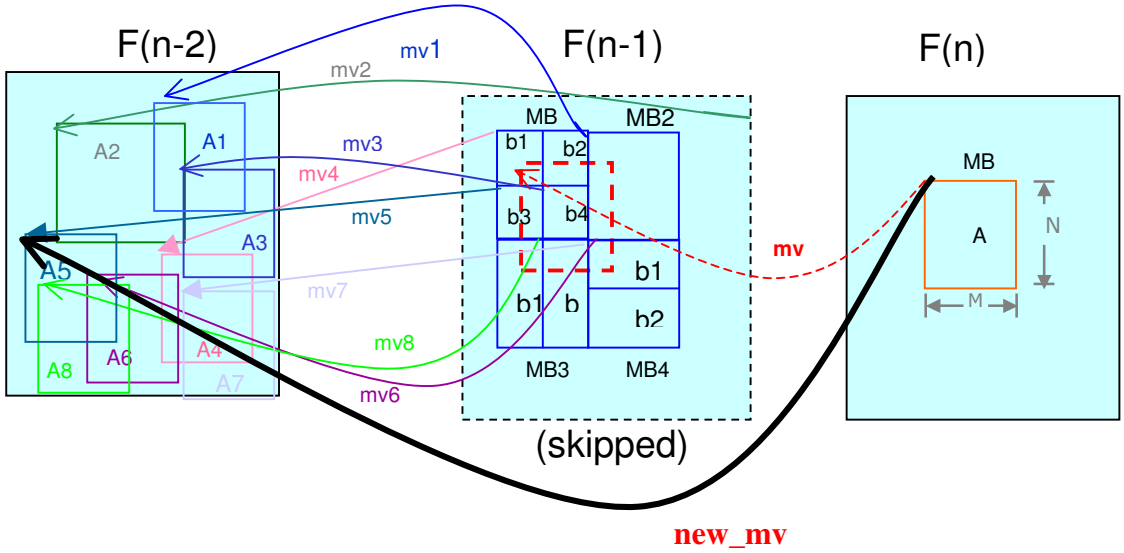


Figure 26. New motion vector composition algorithm

Simulation results show that our algorithm achieves a PSNR comparable to that of total Motion Estimation (ME) process with a great reduction of the computation time (50%), as we show in Figure 27.

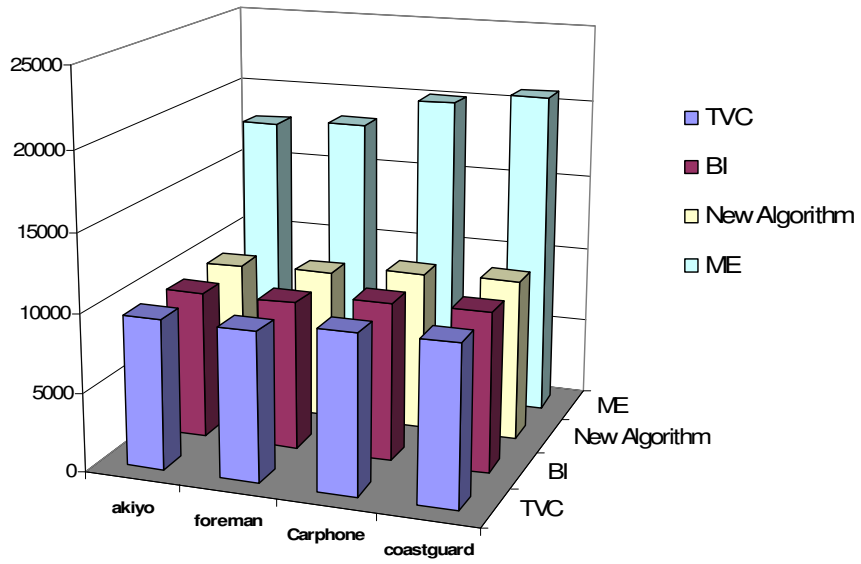


Figure 27. H.264 MVC evaluation

Chapter 6: Future work

The previous survey on frame skipping has shown that many problems are still open, and need to be investigated. We worked on the design of an efficient temporal transcoder for real-time applications in mobile networks. We studied the performance of some frame skipping strategies and Motion Vector Composition algorithms, presented in this report. We observe that, in literature, the frame skipping problem is defined mainly by motion information in an experimental way [39]. It would be interesting to investigate this problem by an analytical approach, by techniques such as randomization and dynamic programming, to design new frame skipping strategies.

The crucial rule that an accurate estimation of the real-time has in many multimedia applications, suggests the importance of investigating on real-time constraints of the transcoding process. We studied the real-time issue in temporal transcoding, and we designed new frame skipping policies able to guarantee a minimum transmission delay. In our performed policies this delay is assumed to be equal to 500 ms, but it is possible to reduce such delay, avoiding also the initial delay due to the first intra frame coding. It would be interesting to perform this by an analytical study of the buffer, validated by an extensive simulation phase. Currently, our policies are tested on MPEG-4 and H.263 temporal transcoders. Extensive results have been obtained in both cases. It is possible to apply our frame skipping policies on temporal transcoder based on the H.264 codec.

In mobile systems, temporal transcoding is a very promising approach to transcode the video sequence with low output bit rate. A problem in temporal transcoding, mentioned in the previous section, is skipping of consecutive frames. It would be interesting to investigate new strategies that minimise the number of consecutive skipped frames, but we think that when a high reduction of the bandwidth occurs (hard transcoding conditions), skipping of consecutive frames at the transcoder is often unavoidable. However, these hard transcoding conditions require a great bit rate reduction, so only traditional methods based on requantisation are not sufficient to produce acceptable image distortion. A potential solution at this problem, is the design of a trade off between temporal and quality transcoding. The goal is to apply requantisation in order to reduce the size of transcoded frame, avoiding consecutive frame skipping in hard transcoding conditions. The motion of the frame could be another important issue to investigate in the combined temporal and quality transcoding approach.

As introduced in the previous sections, most existing transcoding researches are focused on the transcoding algorithm itself, while the transcoded video is always assumed to be

transmitted over a simplified Constant Bit Rate (CBR) channel. An interesting research topic is video transmission over wireless links involving a time-varying channel. In a wireless channel, there is a variable effective channel rate (VBR), due to the burst errors during the channel fading periods. It is possible to use available channel information to modify the transcoder's behaviour according to the channel changes. The main intuition is that the transcoder should reduce the frame rate and/or the frame quality when the effective channel bandwidth is lower. Clearly, the success of these approaches will depend on the existence of channel information control schemes, models of the channel and/or some online observation of its current state.

Beyond the limited available bit-rate, wireless multimedia transmission presents a number of other technical challenges. One of the more difficult issues is the fact that a wireless network cannot provide a guaranteed quality of service, since high bit error rate occurs. Moreover, the temporal and spatial prediction used in video coding standards makes the coded video stream more vulnerable to channel errors. Error-free delivery of data packets can only be achieved by allowing retransmission of lost or damaged packets, through mechanisms such as Automatic Repeat Request (ARQ) protocols [41]. Such retransmission, however, may incur in delays that are unacceptable for some real-time applications. To mitigate the effects of channel errors on the decoded video quality, error-handling schemes must be efficiently applied to the video stream. The video transcoding process can be used to insert error resilient features into the compressed video stream for increasing its robustness. Most error resilient features consist in inserting extra overhead bits into the video stream, which decreases the compression efficiency and requires the allocation of greater bandwidth. At the same time, given the limited computing capability of the mobile devices at the decoding end, some error resilient features, which demand more computation when decoding the video stream, should be avoided. It would be interesting to design an error resilient transcoding solution to increase the robustness of the video stream without sacrificing the video quality and increasing decoder's complexity.

Recently, content aware transcoding techniques have been developed [40]. In some specific video application systems, such as videoconferencing and video surveillance, most of the time, some video objects only are active at any given time, and attract the user attention. The transcoding techniques used in these applications, first identify the area of interest in the incoming video frame by some simple segmentation techniques, and then allocate more bits and more error resilience features to this area. It would be interesting to investigate the impact of these techniques on the efficiency of transcoding process in mobile systems.

An other important issue would be to apply transcoding to a specific network environment, such as vehicular Ad-hoc Networks (VANET), that is an emerging field of MANETs.

Acknowledgements

We thank all ERI people, in particular Roberto Sabella and Emilia Peciola, who introduced us in this research area, Giovanni Iacovoni and Salvatore Morsa for helpful discussions and advices. We thank all Pisatel Lab people, in particular Ing. Antonia Bertolino. Finally, we thank all students who worked with us in this project.

References

[1] S. Dogan, S. Eminsoy, A. H. Sadka and A. M. Kondoz. Video Content Adaptation Using Transcoding for Enabling UMA over UMTS. *Workshop on Image Analysis for Multimedia Interactive Services (WIAMIS '2004)*, Portugal, April 2004.

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

[3] P. Assunção and M. Ghanbari. Optimal Transcoding of Compressed Video. *Proc. IEEE International Conference on Image Processing*, Vol. 1, pp.739-742, October 1997.

[4] P. Assunção and M. Ghanbari. A frequency-domain video transcoder for dynamic bit-rate reduction of MPEG-2 bitstreams. *IEEE Transactions on Circuits and Systems for Video Technology,* Vol. 8 (8), pp. 953-967, December 1998.

[5] S.W. Wu and A. Gersho. Rate-Constrained Optimal Block-adaptive Coding for Digital Tape Recording of HDTV. *IEEE Transactions on Circuits and Systems for Video Technology,* Vol. 1 (1), pp. 100-112, March 1991.

[6] J. Choi and D. Park. A Stable Feedback Control of the Buffer State Using the Controlled Langrange Multiplier Method. *IEEE Transaction on Circuits and Systems for Video Technology,* Vol. 3, pp. 546-558, September 1994.

[7] I. M. Pao, M.T Sun and S.M. Lei. Encoding DCT Coefficients Based on Rate-Distortion Measurement. *Journal of Visual Communication and Image Representation,* Vol. 12 (1), pp. 29-43, March 2001.

[8] Y. Yang and S. S. Hemami. Generalized Rate-Distortion Optimization for Motion-Compensated Video Coders. *IEEE Transactions on Circuits and Systems for Video Technology,* Vol. 10 (6), pp. 942-955, September 2000.

[9] Z. Lei and N. D. Georganas. Rate Adaptation Transcoding for Precoded Video Streams**.** *Proceedings of the tenth ACM international conference on Multimedia*, pp. 127-136, December 2002.

[10] N. Bjork and C. Christopoulos. Transcoder Architectures for Video Coding. *IEEE Transactions on Consumer Electronics*, Vol. 44 (1), pp. 88-98, February 1998.

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

[12] T. Shanableh and M. Ghanbari. Heterogeneous Video Transcoding to Lower Spatio-Temporal Resolutions and Different Encoding Formats. *IEEE Transactions on Multimedia,* Vol. 2 (2), pp. 101-109, June 2000.

[13] G. Shen, B. Zeng, Y.Q. Zhang and M. L. Liou. Transcoder with arbitrarily resizing capability. *IEEE International Symposium on Circuits and Systems,* Vol. 5, pp. 25-28, May 2001.

[14] J. Xin, M. T. Sun, K. Chun and B. S. Choi. Motion Re-estimation for HDTV to SDTV Transcoding. *IEEE International Symposium on Circuits and Systems,* Vol. 4, pp. 715-718, May 2002.

[15] J. N. Hwang, T. D. Wu and C. W. Lin. Dynamic Frame-Skipping in Video Transcoding. *IEEE Transactions on Consumer Electronics,* Vol. 44, pp. 88-98, February 1998.

[16] J. Youn, M. T. Sun and C. W. Lin. Motion Vector Refinement for High-Performance Transcoding. *IEEE Transactions on Multimedia*, Vol. 1 (1), pp. 30-40, March 1999.

[17] T. Shanableh and M. Ghanbari. Heterogeneous Video Transcoding to Lower Spatio-Temporal Resolutions and Different Encoding Formats. *IEEE Transactions on Multimedia*, Vol. 2 (2), pp. 101-110, June 2000.

[18] 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*, Vol. 12 (4), pp. 269-275, April 2002.

[19] K. T. Fung, Y. L. Chan and W. C. Siu. New Architecture for Dynamic Frame-Skipping Transcoder. *IEEE Transactions on Image Processing*, Vol. 11 (8), pp. 886-900, August 2002.

[20] H. Shu and L.P. Chau. Frame-Skipping Transcoding with Motion Change Consideration. *IEEE International Symposium on Circuits and Systems (ISCAS 2004)*, Canada, May 2004.

[21] P. F. Correia, V. M. Silva and P. A. Assunção. A Method for Improving the Quality of Mobile Video under Hard Transcoding Conditions. *Proc. of IEEE International Conference on Communications*, Vol. 26 (1), pp. 928 - 932, May 2003.

[22] T. Koga, A. Iinuma, Y. Iijma, and T. Ishiguro. Motion-compensated interframe coding for video conferencing. *Proc. of National Telecommunications Conference*, pp. G5.3.1–G5.3.5, November 1981.

[23] S. F. Chang and D. G. Messerschmitt. A New Approach to Decoding and Compositing Motion compensated DCT-Based Images. *IEEE International Conference on Acoustics, Speech, and Signal Processing,* Vol. 5, pp. 421-424, April 1993.

[24] P. Assunção and M. Ghanbari. Transcoding of MPEG-2 Video in the Frequency Domain. *IEEE International Conference on Acoustics, Speech, and Signal Processing,* Vol. 4, pp. 2633-2636, April 1997.

[25] H. Sun, W. Kwok and J. W. Zdepski. Architectures for MPEG Compressed Bitstream Scaling. *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 6 (2), pp. 191-199, April 1996.

[26] Y. Nakajima, H. Hori and T. Kanoh. Rate Conversion of MPEG Coded Video by Re-quantization Process. *IEEE International Conference on Image Processing,* Vol. 3, pp. 408-411, October 1995.

[27] K. S. Kan, K. C. Fan and Y. H. Huang. Low-Complexity and Low-Delay Video Transcoding for Compressed MPEG-2 Bitstream. *IEEE International Symposium on Consumer Electronics*, Vol. 2 (4), pp. 99-102, December 1997.

[28] N. Merhav and V. Bhaskaran. A Transform Domain Approach to Spatial Domain Image Scaling. *HP Labs Technical Report,* HPL 94-116, December 1994.

[29] W. J. Lee and W. J. Ho. Adaptive frame-skipping for video transcoding. *Proc. of International Conference on Image Processing*, Vol. 1, pp.165-168, September 2003.

[30] J. Youn, M. T. Sun and C. W. Lin. Motion estimation for high performance transcoding. *IEEE Transactions on Consumer Electronics*, Vol. 44 (3), pp. 649-658, August 1998.

[31] K. T. Fung, Y.L. Chan and W. C. Siu. Low-complexity and high-quality frame-skipping transcoder. *IEEE International Symposium on Circuits and Systems*, Vol. 5, pp. 29-32, May 2001.

[32] 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*, Vol. 6 (1), pp.31- 46, February 2004.

[33] P. F. Correia, V. M. Silva and P. A. Assunção. Rate Prediction Model for Video Transcoding Applications. *IEEE International Symposium on Telecommunications*, Vol. 1, pp. 641-644, September 2002.

[34] H. Lee, T. Chiang and Y. Zhang. Scalable Rate Control for MPEG-4 Video. *IEEE Transactions on Circuits and Systems for Video Tecnology*, Vol. 10 (6), pp. 878-894, September 2000.

[35] K. Seo, S. Kwon, S. K. Hong and J. Kim. Dynamic Bit-rate reduction Based on Frame skipping and Requantization for MPEG-1 to MPEG-4 Transcoder. *Proc. of IEEE International Symposium on Circuits and Systems,* Vol. 2, pp. 372-375, May 2003.

[36] I. E.G Richardson. H.264 and MPEG-4 Video Compression, Video Coding for Next-generation Multimedia. Wiley 2003.

[37] Telenor Research (TR). Video Codec Test Model: TMN5. *ITU Telecommunication Standardization Sector LBC - 95, Study Group 15, Working Party 15/1*, January 1995.

[38] P. Assunçao and M. Ghanbari. Congestion control of video traffic with transcoders. *IEEE International Conference on Communications*, Vol. 1, pp. 523-527, June 1997.

[39] F. Pan, Z. P. Lin, X. Lin, S. Rahardja, W. Juwono and F. Slamer. Content adaptive Frame skipping for low bit rate video coding. *Proc. of IEEE International Conference on Multimedia Information, Communications and Signal Processing,* Vol. 1, pp. 230-234, December 2003.

[40] J. I. Khan, Z. Guo and W. Oh. Motion based Object Tracking in MPEG-2 Stream for Perceptual Region Discriminating Rate Transcoding**.** *Proc. of the ninth ACM International Conference on Multimedia,* Vol. 9, pp. 572-576, June 2001.

[41] J. Kurose and K. Ross. Computer Networking: A Top Down Approach Featuring the Internet. Addison-Wesley, July 2002.

[42] http://megaera.ee.nctu.edu.tw/mpeg/Optimized_Ref_Software/

[43] TMN encode-decode, Version 3.0, May 27, 1997 (C) Department of Electrical Engineering University of British Columbia CANADA. Michael Gallant,Guy Cote,Berna Erol,Faouzi Kossentini (Based on Telenor's tmn version 2.0).

[44] VIDEO CODING FOR LOW BIT RATE COMUNICATION, ITU-T Recommendation Draft H.263, version 20, 16 Sept. 1997.

[45] K. Ramchandran, A. Ortega, and M. Vetterli. Bit allocation for dependent quantization with applications to multiresolution and MPEG video coders, *IEEE Transactions on Image Processing*, vol. 3, pp. 533-545, Sept. 1994.

[46] VIDEO CODEC TEST MODEL, NEAR-TERM, VERSION 5 E 8 (TMN5, TMN8) RELEASE 0 ITU - Telecommunications Standardization Sector – Video Coding Experts Group, Portland 24-27 June 1997.

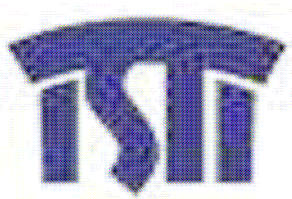
[47] Z. He, S. K. Mitra. Optimum bit allocation and accurate rate control for video coding via ρ-domain source modelling. *IEEE Transactions on circuits and systems for video technology*, vol.12 no. 10, Oct. 2002.

[48] C.W. Wong, O. C. Au, B. Meng, H. K. Lam. Perceptual rate control for low-delay video communications. *IEEE Multimedia and Expo*, ICME vol.3, Jul. 2003.

[49] http://iphome.hhi.de/suehring/tml/download/

[50] STUDY OF FINAL COMMITTEE DRAFT OF JOINT VIDEO SPECIFICATION (ITU-T REC.H.264—ISO/IEC 14496-10 AVC), Draft 2.

[51] T. Wiegand, G.J Sullivan, G. Bjntegaard, A. Luthra. A Overview of the H.264/AVC video coding standard. *IEEE Transactions on circuits and systems for video technology*, vol. 13 no. 7, July 2003.



PISATEL

VIDEO TRANSCODING TECHNIQUES IN MOBILE SYSTEMS

Prof. Maurizio Bonuccelli

Francesca Martelli

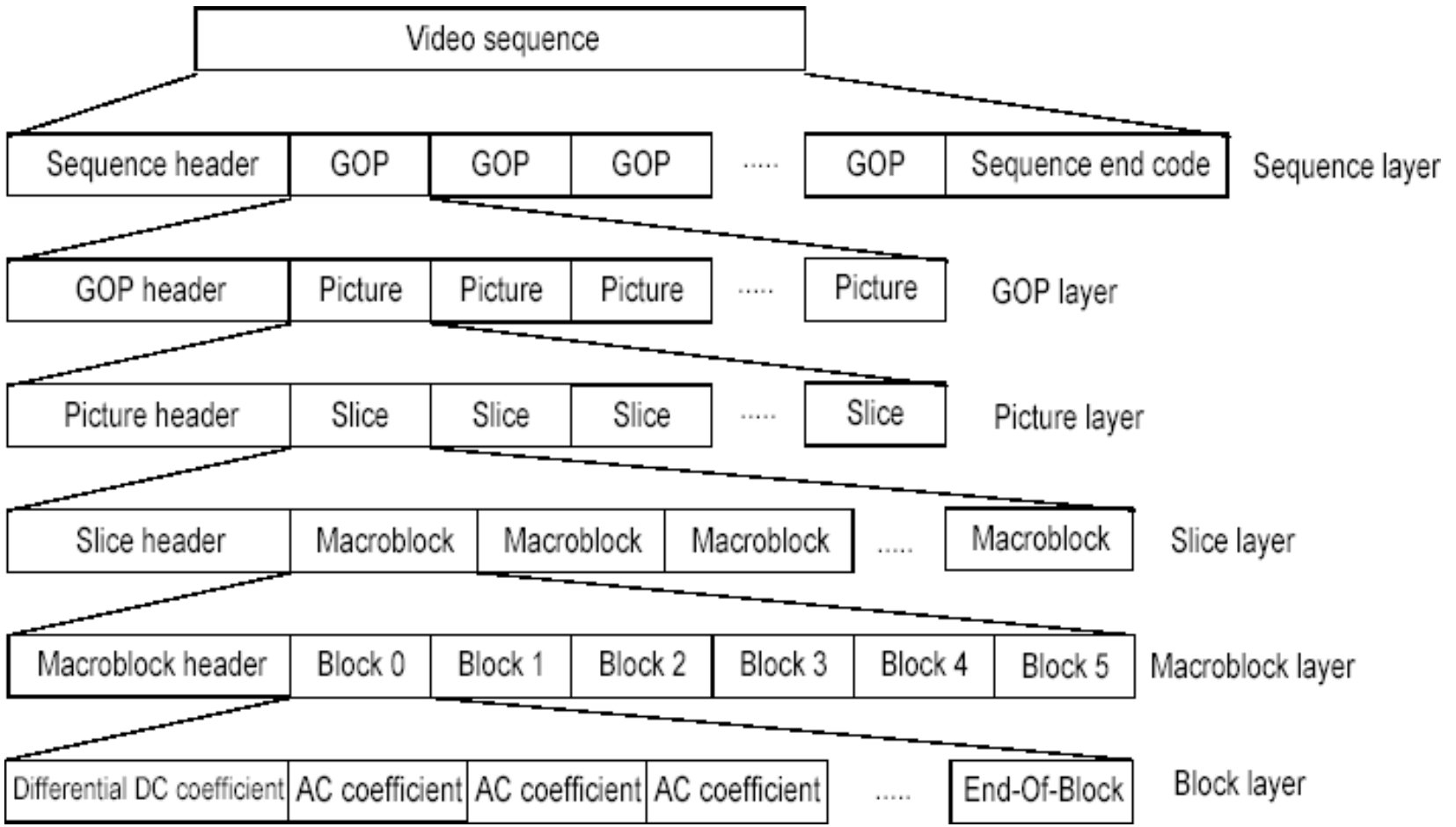
Francesca Lonetti

PISATEL

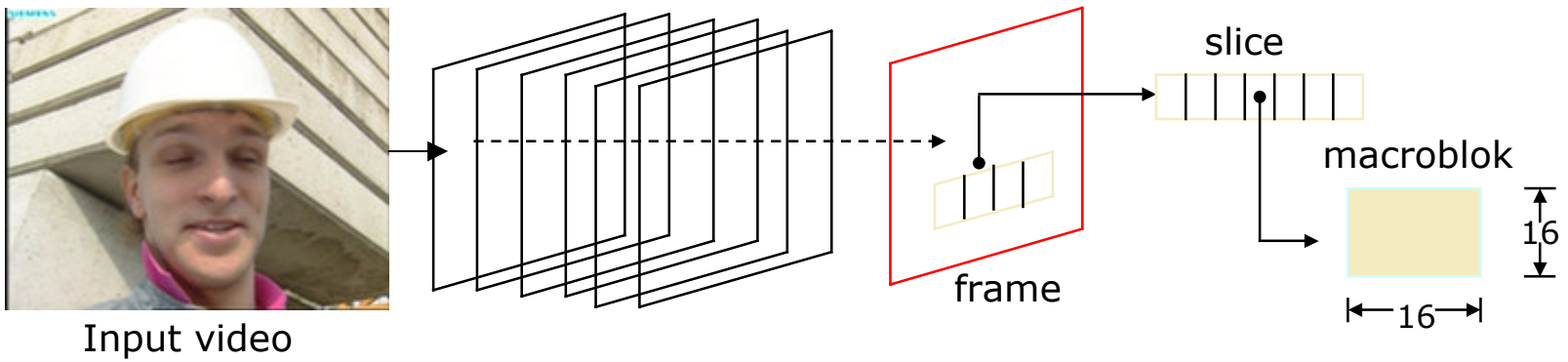
Overview

- Video coding features
- Transcoding in third generation mobile communication systems
- Temporal transcoding
- Motion Vector Composition Algorithms
- Frame Skipping Policies
- Quality vs Temporal transcoding
- H.264 coding and transcoding
- Conclusion

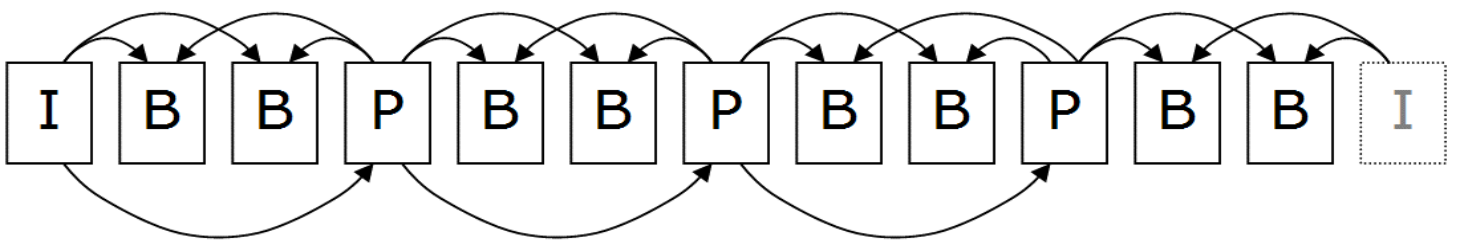
Coding a Video Sequence



Video Coding



- I frames (intra-frames)
- P frames (forward predicted frames)
- B frames (bi-directional predicted frames)



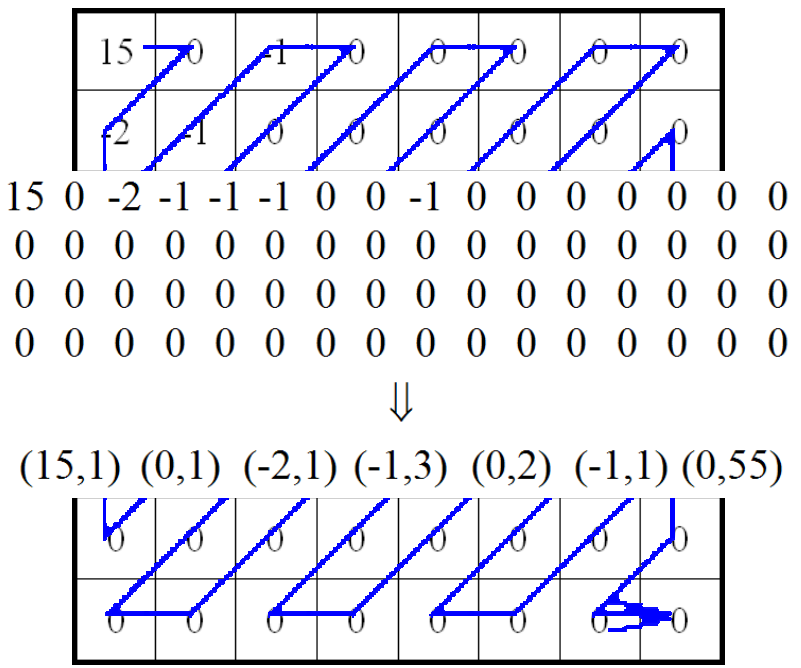
Video Compression

- Most coding video standards are based on the same hybrid framework (DCT/MCP)
  - Intraframe coding: spatial redundancy within a frame is removed by DCT (Discrete Cosine Transform)
  - Interframe coding: temporal redundancy among frames is removed by MCP (Motion Compensated Prediction)

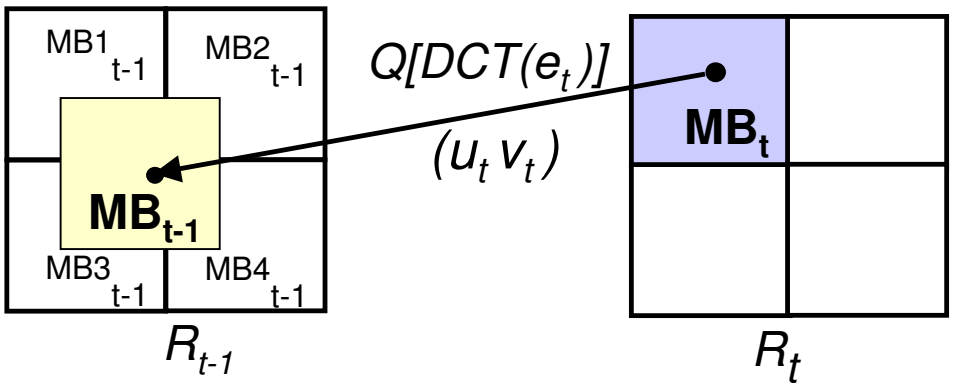
Intraframe Coding

It consists of the following steps:

1. DCT
2. Quantization
3. Entropic coding:
   - Zig-Zag Scan
   - Run Length Coding
   - Variable Length Coding



Interframe Coding

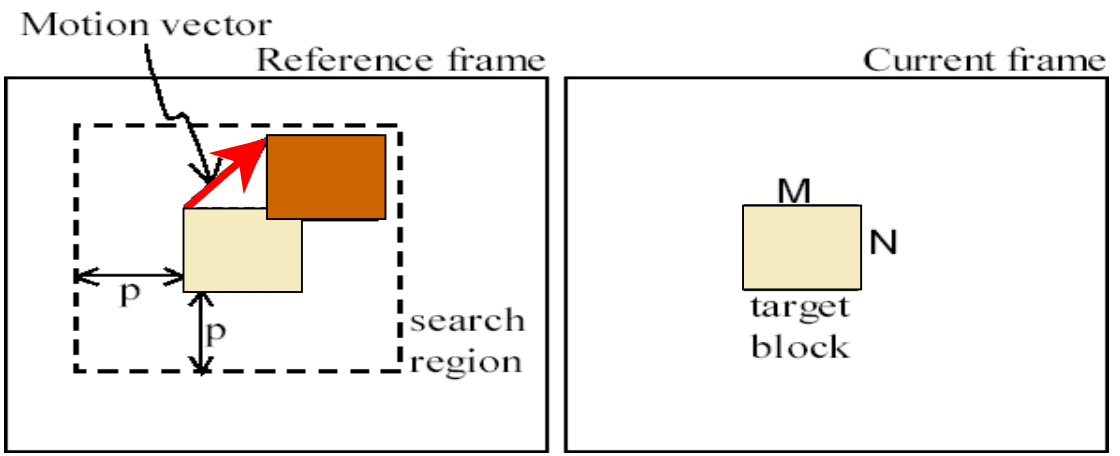


It consists of coding only the differences with a reference frame:

- Motion Vector: indicates the reference frame area most similar to the macroblock to code
- Prediction Errors: the differences between the best match area and the macroblock are coded in the same manner of the intraframe coding

Motion Estimation

- It searches the best match of a macroblock in the reference frame within a search region
- The best match is that one having the minimum SAD (Sum of Absolute Differences) value



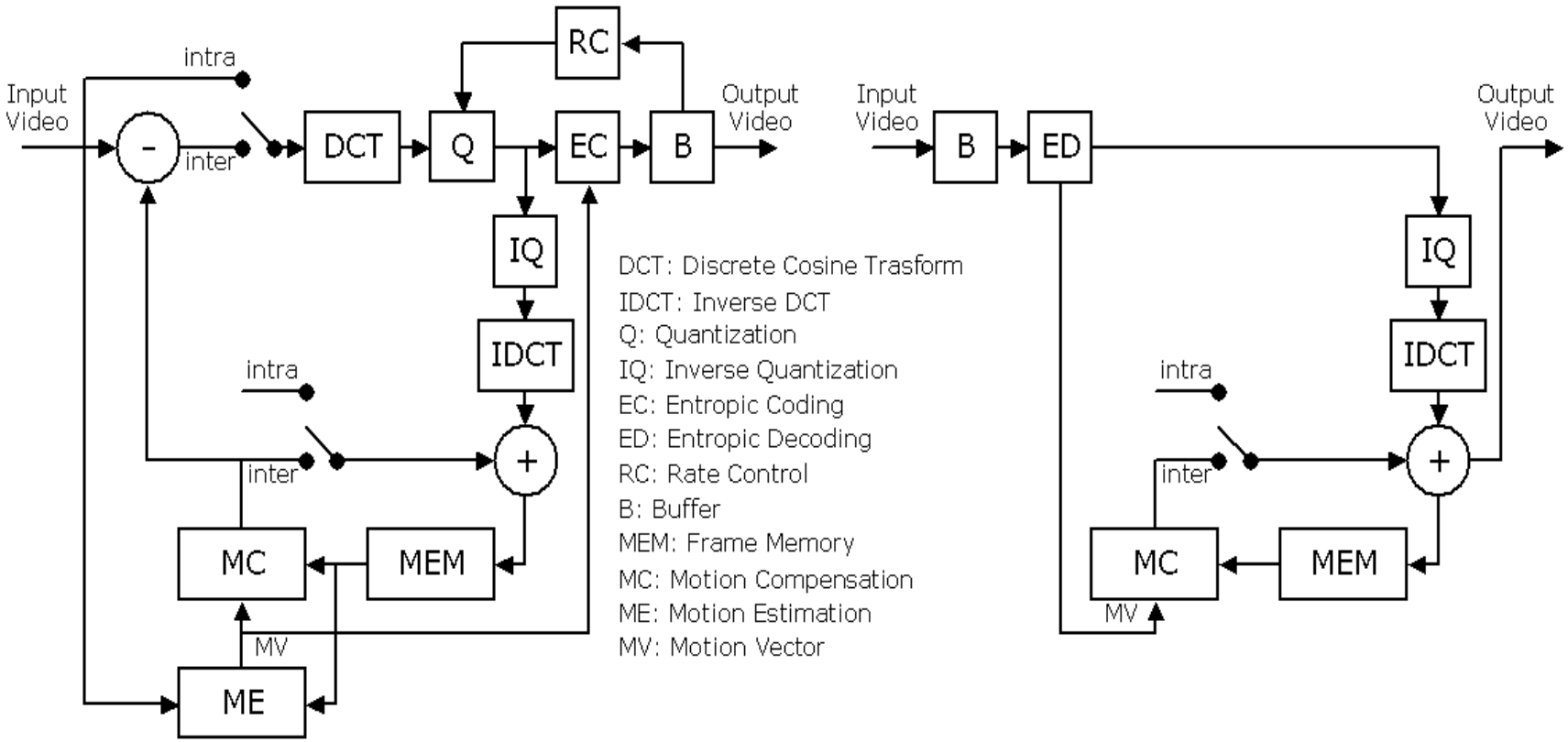
Motion Estimation Algorithms

- The Full Search algorithm computes the SAD values of all possible matching areas in the search region
- It is the most computational complex motion estimation algorithm (it requires about 60-70% of video compression total time)
- There are other faster algorithms (Three-Step Search)

Rate Control

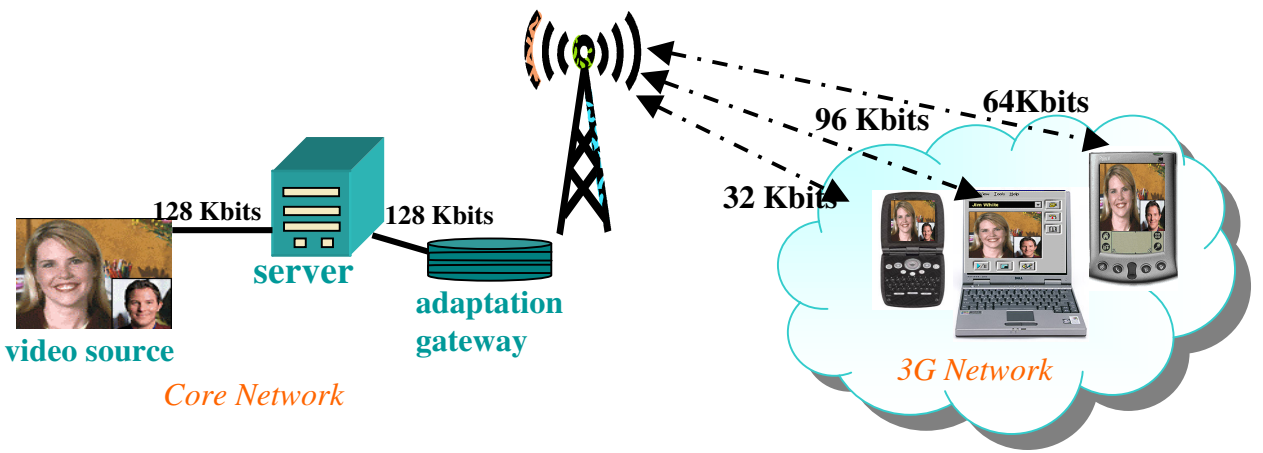
- By using fixed quantization parameters, the compressed video sequence has a variable bit rate; this may cause buffer underflow or overflow
- The rate control concerns about the computation of quantization parameters in order to adapt the video streaming bit rate with the channel bandwidth
- It computes the complexity estimation of the current frame by using that one of previous frame

Standard DCT/MCP Video Codec



Video Transcoding

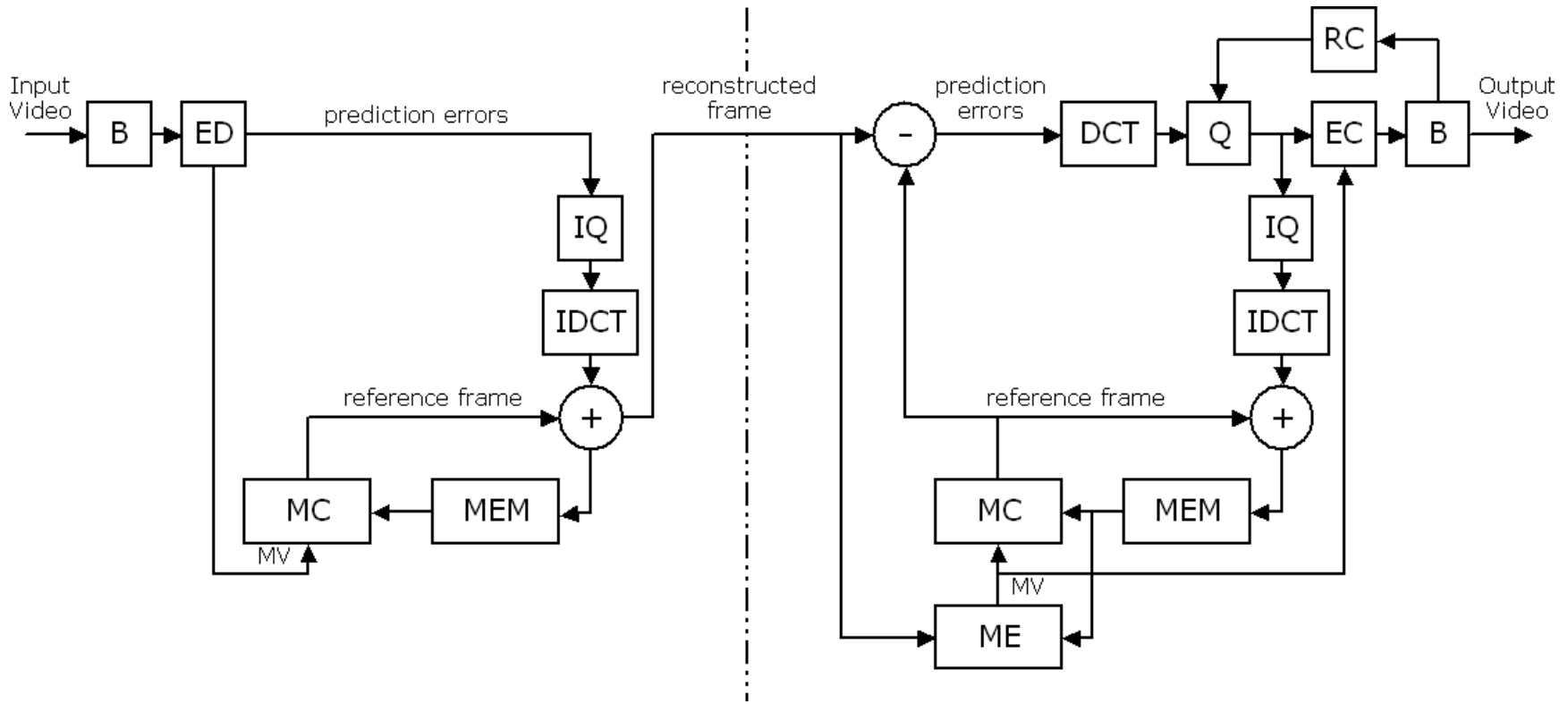
- Transcoding converts a coded video sequence into another one with:
  - Different format
  - Different frame resolution (spatial transcoding)
  - Different video quality (quality transcoding)
  - Different frame rate (*temporal transcoding*)
- Advantages:
  - Interoperability of heterogeneous mutimedia networks



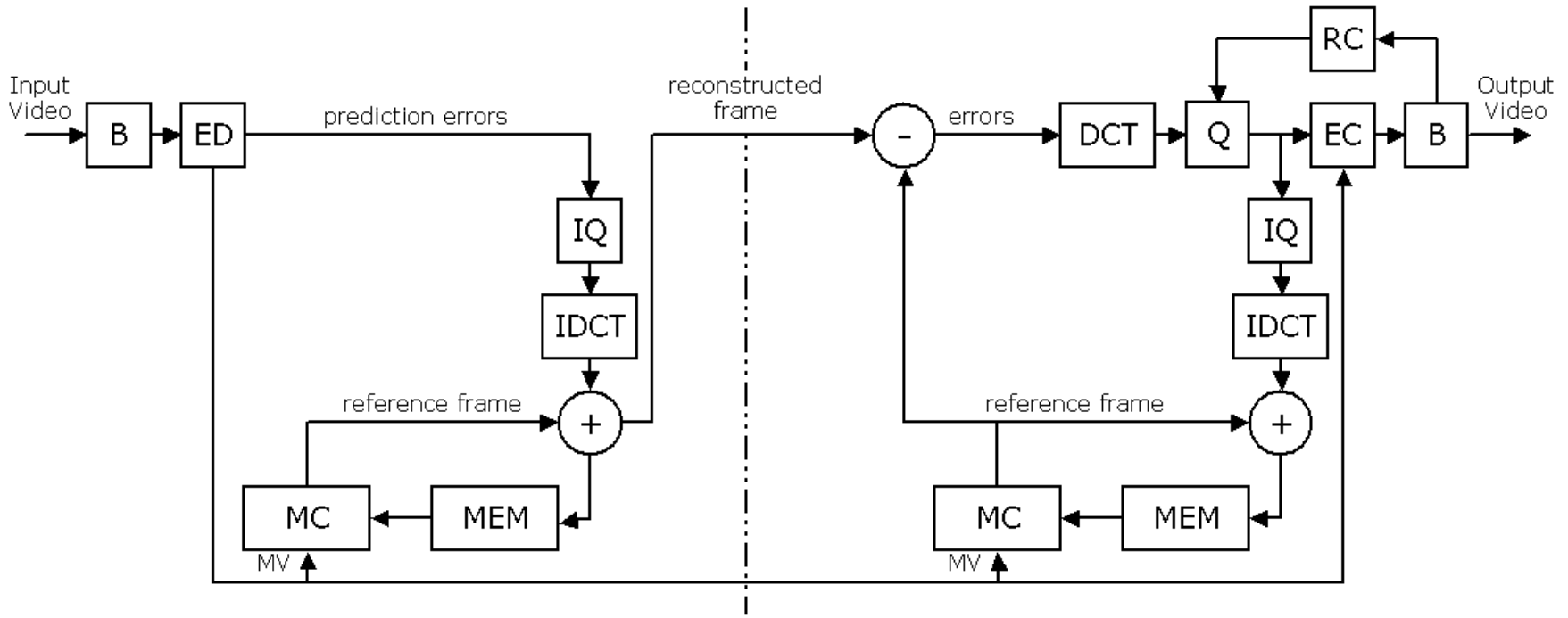
Video Transcoding Architectures

- There are essentially three types of video transcoding architectures:
  - Pixel Domain Transcoder
  - DCT Domain Transcoder
  - Open-Loop Transcoder

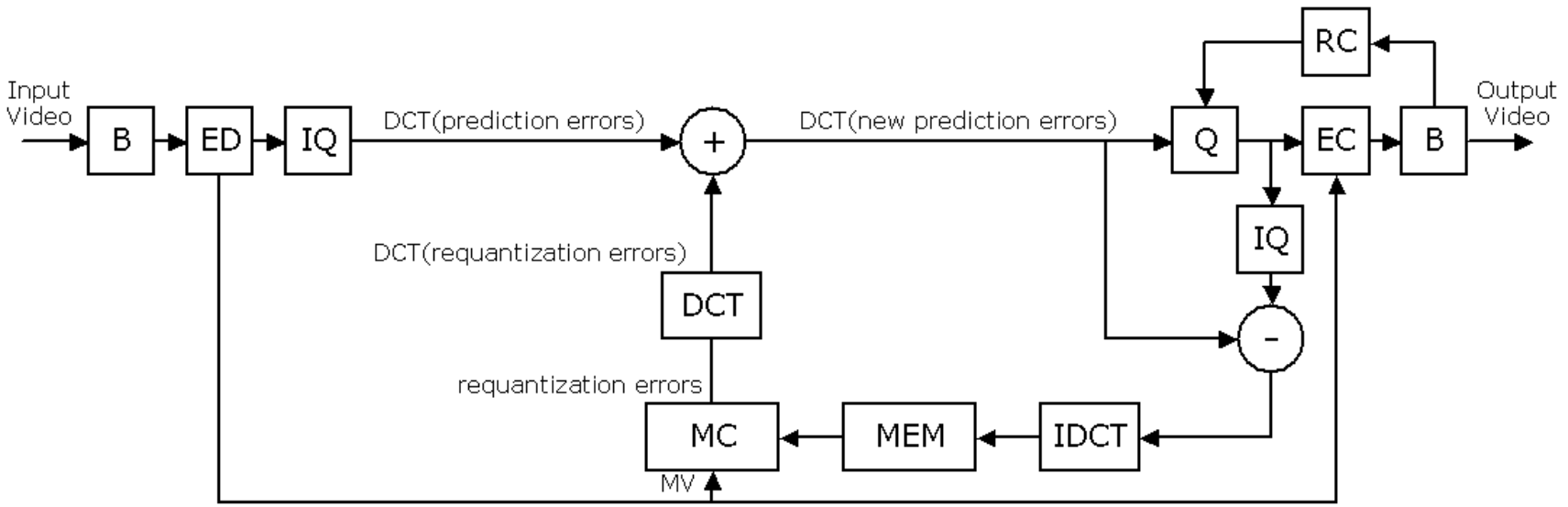
General Transcoder Architecture



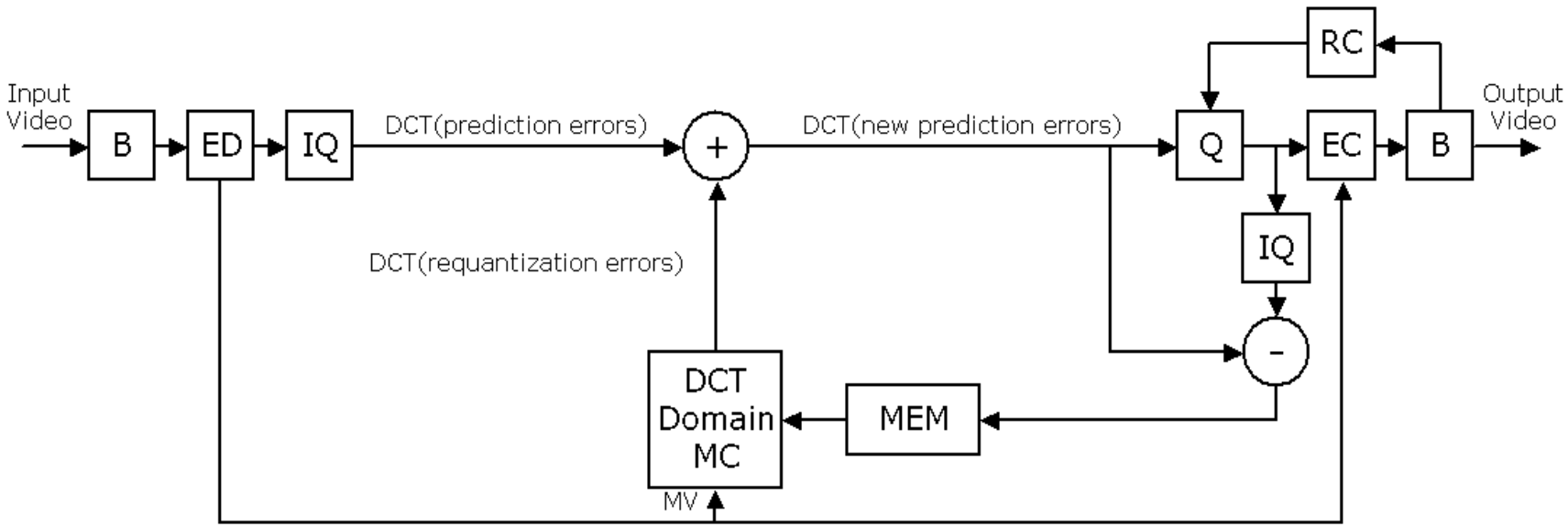
Cascaded Pixel Domain Transcoder



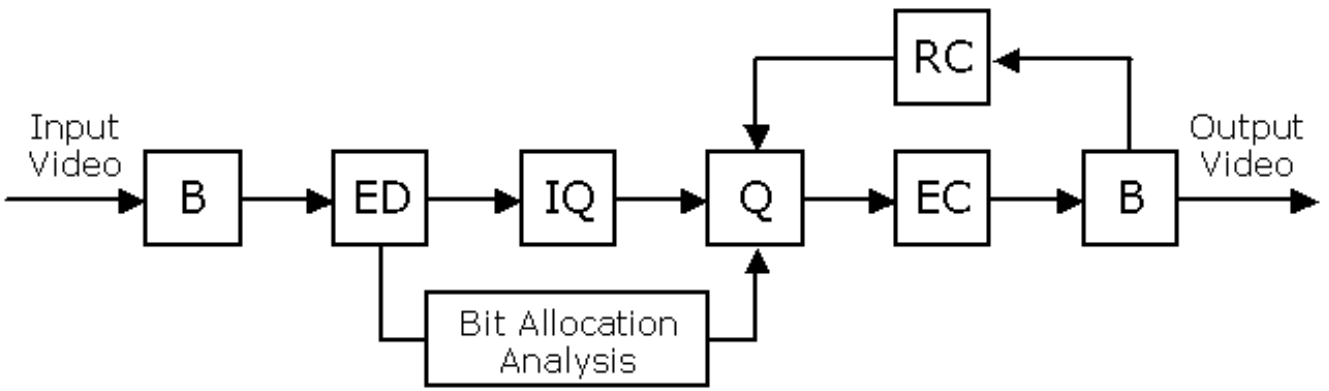
Fast Pixel Domain Transcoder



DCT Domain Transcoder



Open-Loop Transcoder



Spatial Transcoding

- It consists of reducing the frame resolution of the input video stream
- There are two types of spatial transconding:
  - Subsampling 2:1
  - Arbitrary sampling
- In both approaches it is necessary to recompute the motion vectors and the prediction errors by combining and scaling the original ones

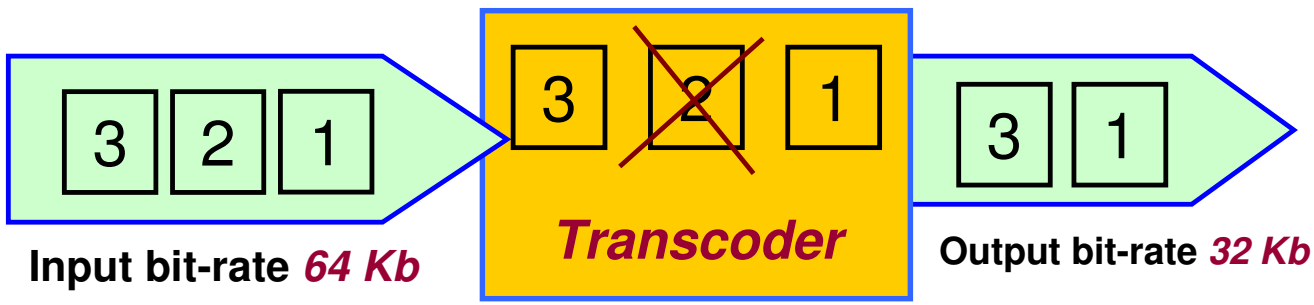
PISATEL

Quality Transcoding

- It consists of decreasing the video stream bit rate by reducing the video quality without changing the frame rate or the frame resolution
- It can be performed by a specific Rate Control function, where the complexity of the current frame, instead of being estimated, can be directly extracted from the input video

Temporal Transcoding

- Skipping frames to reduce the output bit-rate

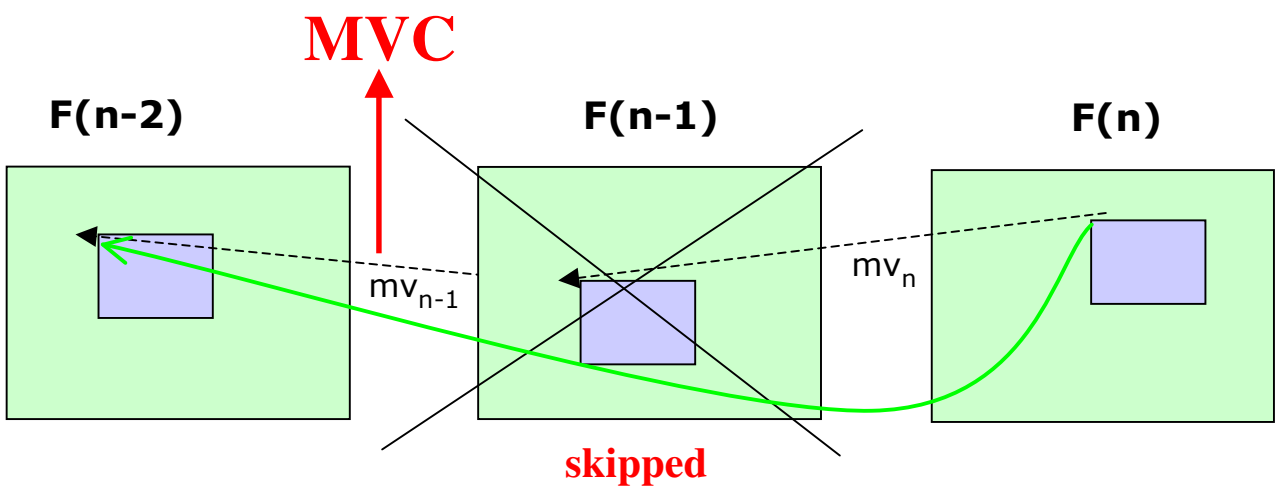


Then it is needed:

- to recompute the motion vectors not still valid, because they point to discarded frames (*Motion Vector Composition*)
- to recompute the prediction errors
- to choose the frames to be skipped (*frame skipping policy*)

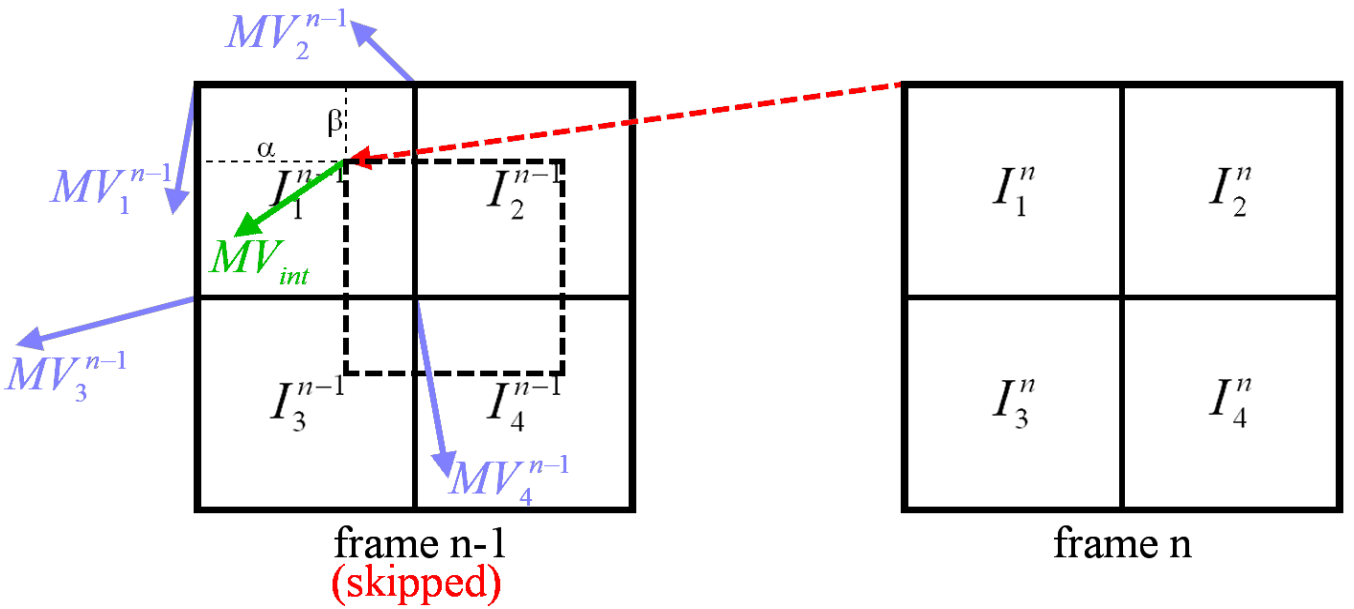
Motion Vector Computation

- The motion vectors are computed by
  - Motion Vector Composition Algorithms (BI, TVC, FDVS, ADVS)
  - Restricted Motion Estimation (RME)



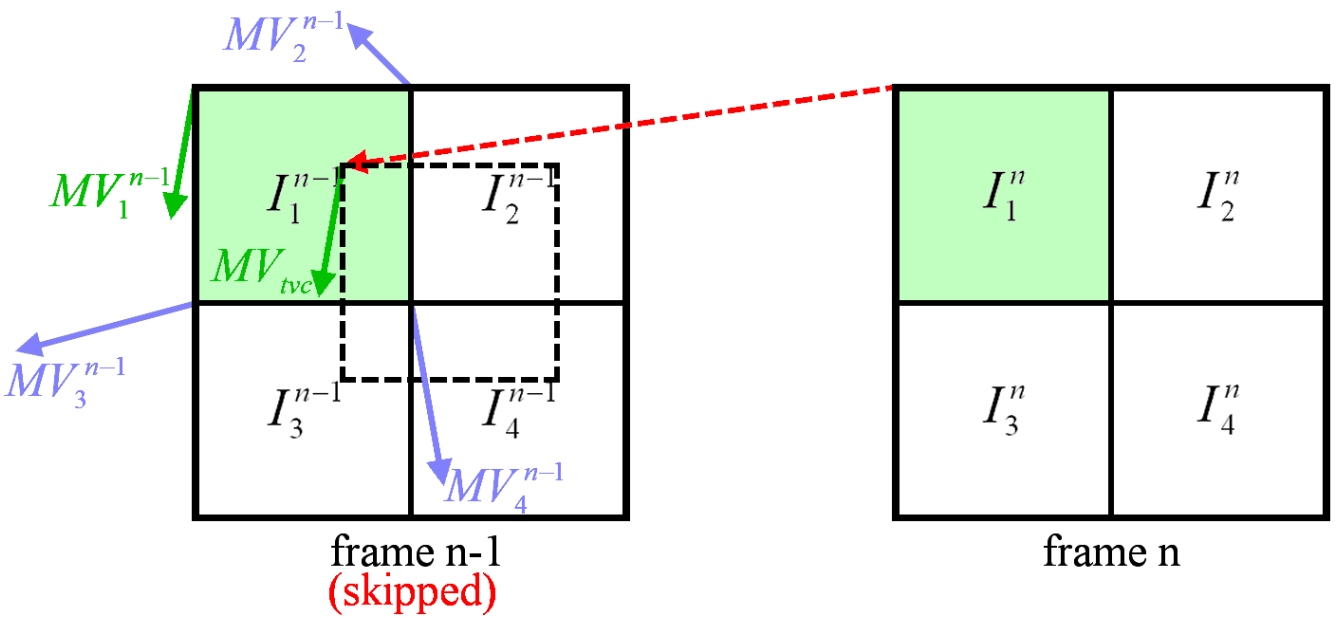
Bilinear Interpolation

$$MV_{int}=(1-\alpha)(1-\beta)MV_1+\alpha(1-\beta)MV_2+(1-\alpha)\beta MV_3+\alpha\beta MV_4$$



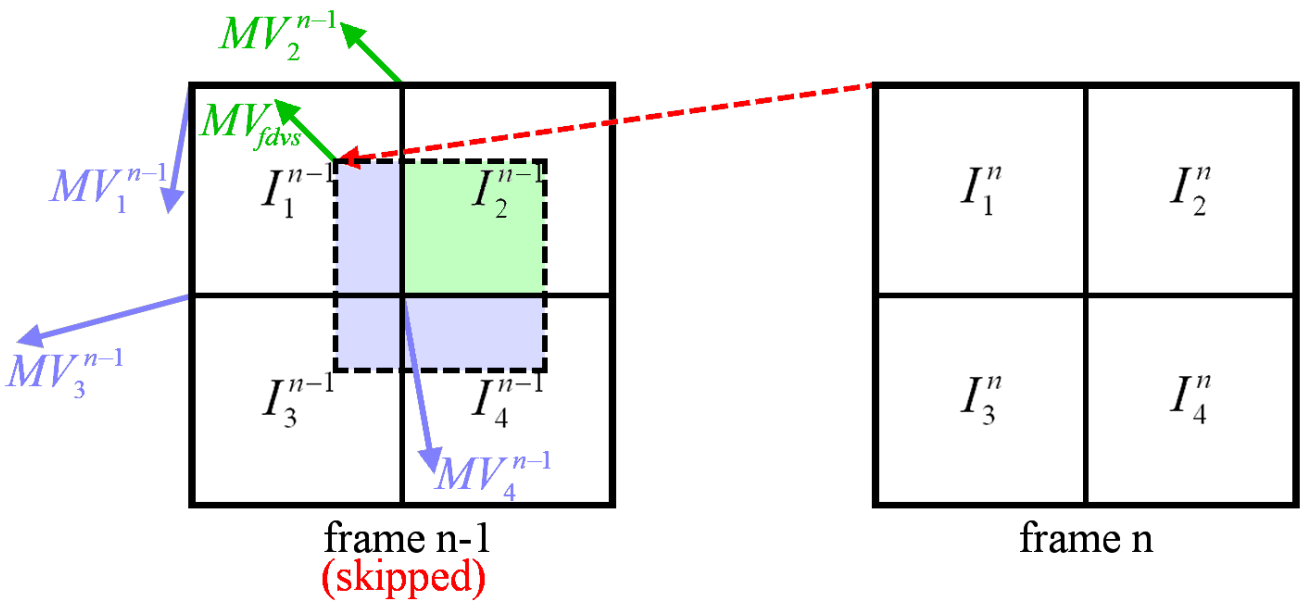
Telescopic Vector Composition

$$MV_{tvc}^{n} = MV_{t}^{n-1}$$



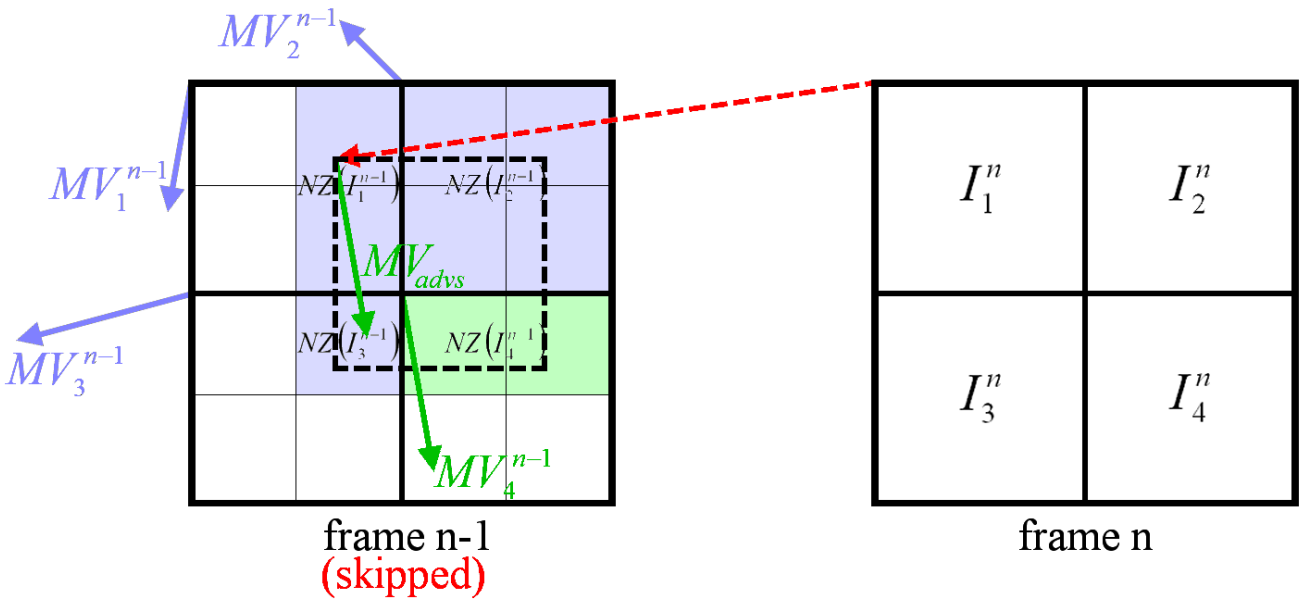
Forward Dominant Vector Selection

- Select the motion vector of the macroblock having the largest intersection area

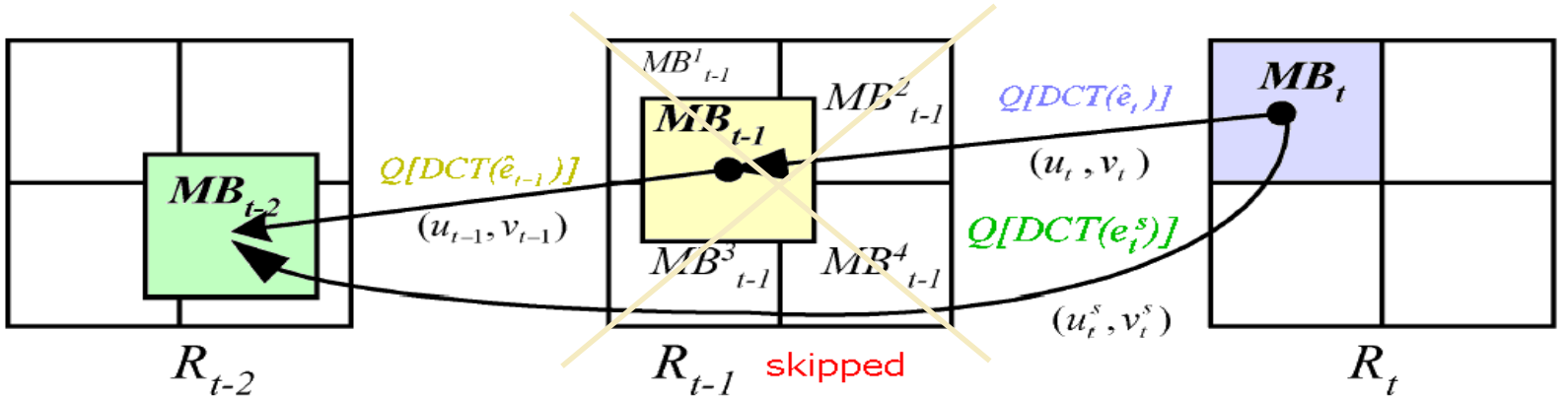


Activity Dominant Vector Selection

- Select the motion vector of the macroblock with the highest number of non-zero quantized DCT coefficients



Dynamic Frame Skipping (DFS)



- Prediction errors: according to the new motion vectors difference between current macroblock and reference area in the not skipped frame is computed, coded with usual DCT and quantization
- Frame skipping policy:
  - $MA_t = \sum_m MA_m$ > Threshold $\Rightarrow$ frame should not be skipped
  - Threshold = *MA* of the previous frames/Number of transcoded frames

Frame Skipping Control (FSC)

- Prediction errors: $Q[DCT(e_t^s)] = Q[DCT(e_t)]+Q[DCT(e_{t-1})]$
- It is needed to recompute $Q[DCT(e_{t-1})] \Rightarrow$ additional error of re-encoding (*RE*)

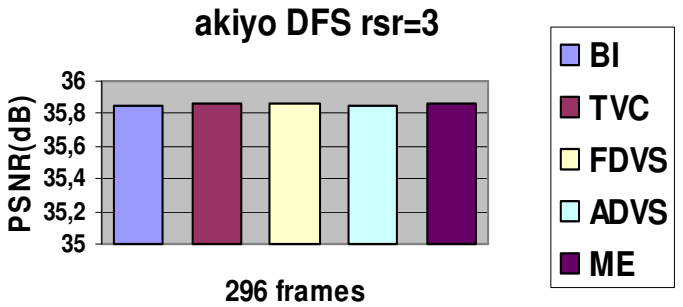
- Frame skipping policy:

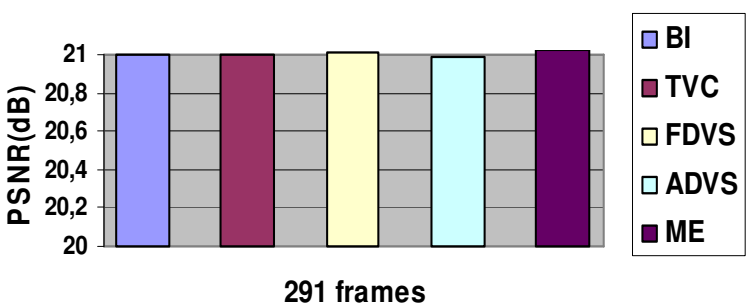
$$FSC_t\,(MA_t\,, RE_t\,) = \frac{\sum_{m=1}^{M} (MA_t\,)_m}{\sum_{m=1}^{M} (RE_t\,)_m} > \text{Threshold} \Rightarrow \text{frame should not be skipped}$$

- Threshold = *MA* of the previous frames/Number of transcoded frames

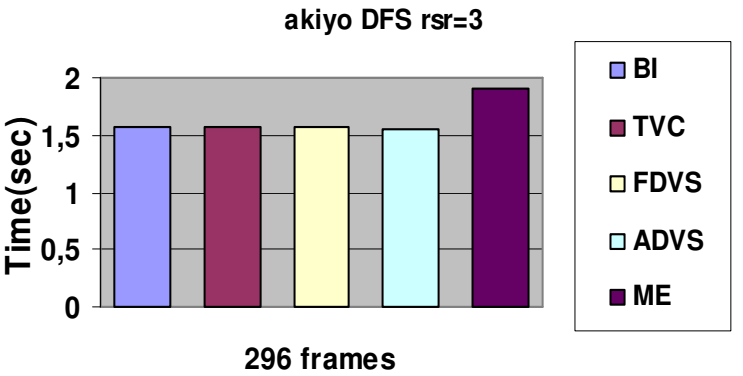
MVC Algorithms Evaluation

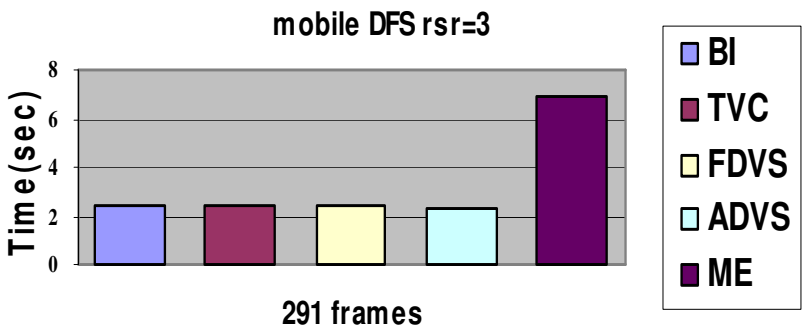
PSNR



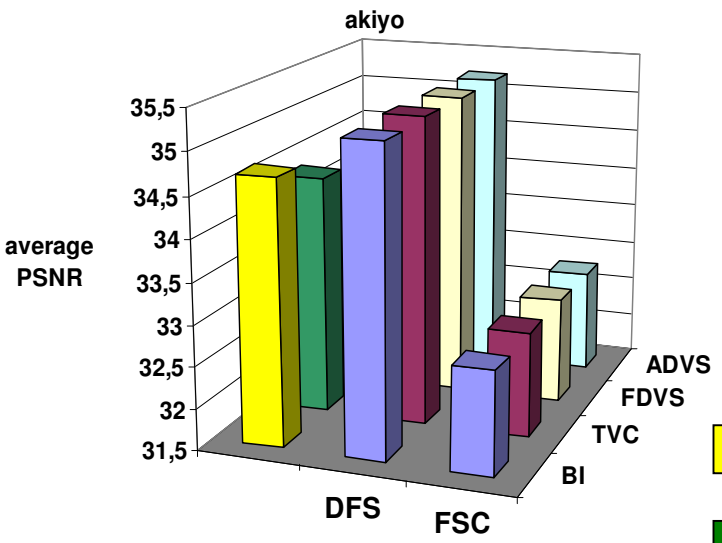


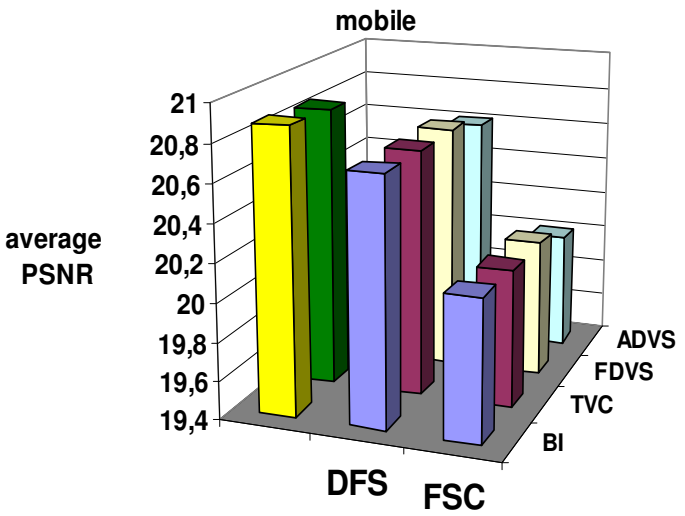
Time





Quality vs Temporal Transcodig (MPEG4)





Q2 ME FS

Q2 ME MVFAST

DFS is better!

Our Temporal Transcoding Architecture

- The motion vectors are computed by
  - Motion Vector Composition Algorithms (BI, TVC, FDVS, ADVS)
  - Restricted Motion Estimation (RME)
- The prediction errors are computed in the pixel domain
- A transcoder output buffer is introduced for facing the real-time problem
  - A basic buffer-based frame skipping policy is developed
  - Other frame skipping policies are developed to improve the video quality

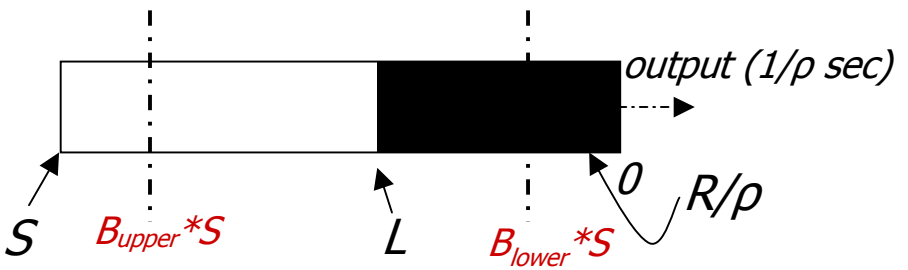
Our goals

- Appliyng frame skipping to reduce output bit-rate
- Guarantee real-time constraints avoiding:
  - Buffer underflow (stop of video sequence)
  - Buffer overflow (increase of delaly, frame loss)
- We define:
  - *IR = input bit rate*
  - *R = output bit-rate*
  - *S = buffer size*
  - *L= buffer occupancy*
  - *ρ = frame rate*
- Two buffer thresholds (B_{lower} and B_{upper}) are defined

Basic skipping policy

- A frame is transcoded if
  - It is the first frame
  - Buffer occupancy <= $B_{lower}*S$
- A frame is skipped if
  - Buffer occupancy > = $B_{upper}*S$
  - The size of the transcoded frame exceeds the free buffer space
- the delay τ is determined by L/R
- If the buffer size is half of the output bit rate the maximum admitted delay is 500 *ms*
- $\tau + \tau_0$ if the first frame > S
- B_{lower} and B_{upper} are dynamically set according to the ratio IR/R

Basic Policy*(frame f)*
if *(f=first frame) transcode f*
else
if *((L<= $B_{lower}(S)$)&(L+ l(f)<=S)) transcode f*
else
if *((L>= $B_{upper}(S)$) skip f*
else
if *(L+l(f)>=S) skip f*
else *transcode f* ***OR apply*** *another policy*



Motion activity based Frame Skipping

- A frame is transcoded if
  - It has large value of *MA*
- A frame is skipped if
  - It has small value of *MA*
- A new motion activity *(MA)* measure is introduced considering different types of motion
- Intra macroblocks have the maximum motion activity value
- The threshold *Thr(f)* is dynamically set to take into account *MA* of the previous transcoded frame and *MA* of all earlier frames

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) <=Thr(f)) skip f*
else *transcode f*

$$MA = \sum_{m} k^{|x_m|} + k^{|y_m|}$$

Random based Frame Skipping

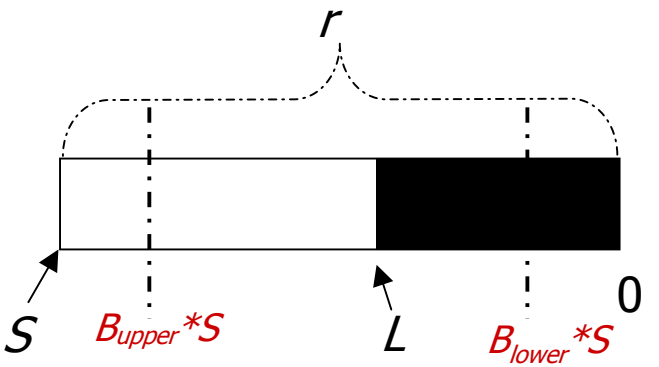
- A random number *r* is uniformly generated in the range [0…S]
- The frame is transcoded when the random number is larger than the buffer occupancy
- Greater is the buffer occupancy, smaller is the probability of transoding the frame

Random Policy *(frame f)*

randomNumber = random() % S

if *(randomNumber >=L) transcode f*

else *skip f*



Consecutive Frame Skipping

- Hard transcoding conditions
  - If the ratio IR/R is high many consecutive frames are skipped
- By skipping many consecutive frames, the size of the transcoded ones increases until to reach an irreversible situation
- goal: distributing uniformly skipped frames
- Forcing trascoder to skip a frame to prevent that many frames are dropped later

How many frames ?

Skipping Γ-1 consecutive frames

Consecutive skipping policy *(frame f)*
if *(numConsecutiveSkipppedFrames < Γ) skip f*
numConsecutiveSkipppedFrames++
else *transcode f*
numConsecutiveSkipppedFrames=0

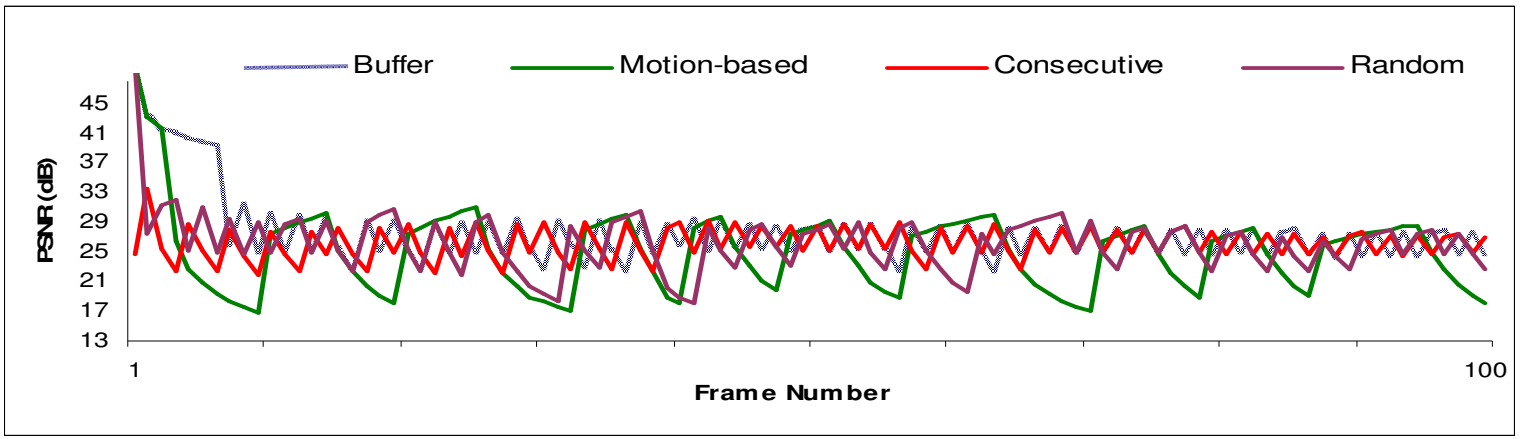
Γ=IR/R N=number of frames

*N*1/*Γ ideal number of transcoded frames

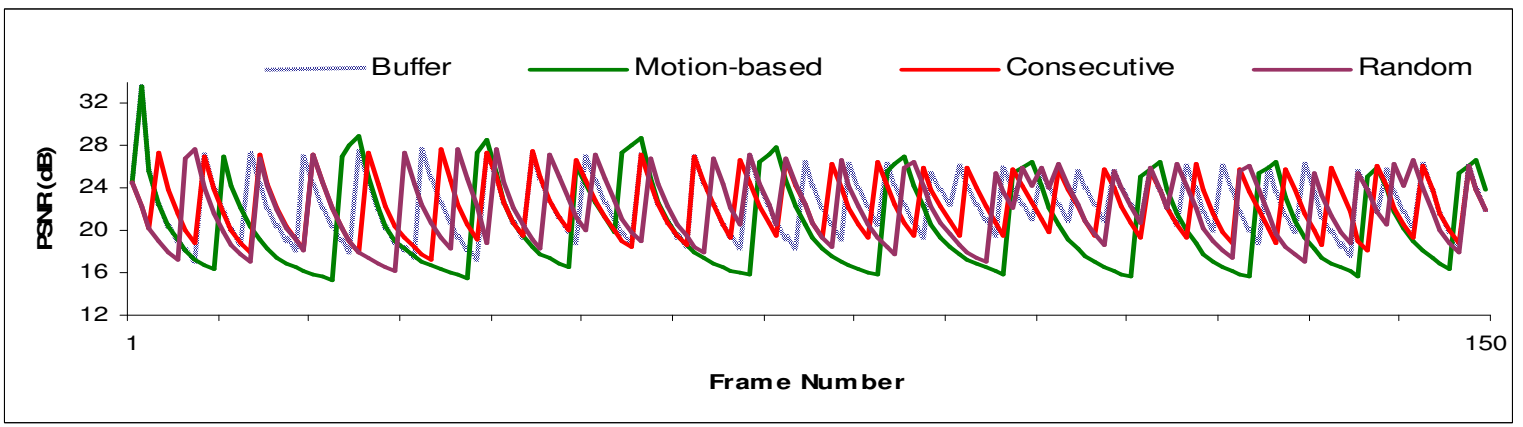
N(1-1/*Γ) ideal number of skipped frames

Frame Skipping policies evaluation (MPEG4)

Mobile video sequence IR=128, R=64



Mobile video sequence IR=128, R=32



Frame Skipping policies evaluation (MPEG4)

	mobile			foreman			coastguard		
	Frames	PSNR1	PSNR2	Frames	PSNR1	PSNR2	Frames	PSNR1	PSNR2
Standard transcoding conditions (IR=128, R=64 kbps)									
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 conditions(IR=128, R=32 kbps)									
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

Frame-Size Prediction

- The size of a transcoded frame increases according to the logarithm of the number of the previously skipped frame by this law

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

- where
  - $l(f)$ is the size of the frame transcoded after skipping f consecutive frames
  - α is a constant proportional to the size of the first skipped frame

Size-Prediction Skipping Policy

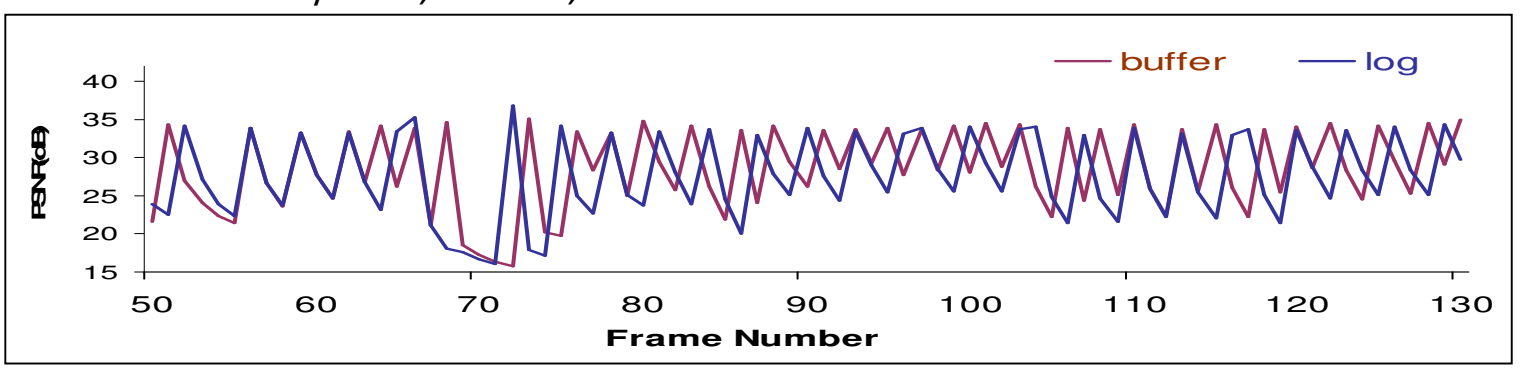
- Predicts the size of the next frame if the current one is skipped
- The frame is transcoded only when its predicted size is lower than the free buffer space
- Advantage: avoids the computation needed to transcode the frames that will be skipped

```
Size-Prediction Policy (frame f):
if (f = first frame) then transcode f
      else
      if ((L ≤ Blower(S)) & (L + l(f) ≤ 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/ρ
          while ((L > Blower(S)) & (L + l(f) ≥ S)  transcode f
```

Size-prediction policy evaluation (H.263)

	Buffer-based				Size prediction			
	frames	PSNR1	PSNR2	Time (sec)	frames	PSNR1	PSNR2	Time (sec)
IR=256, 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, 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

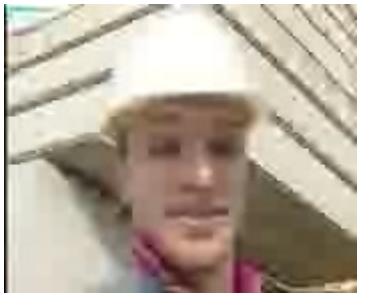
Foreman video sequence, IR=256, R=128



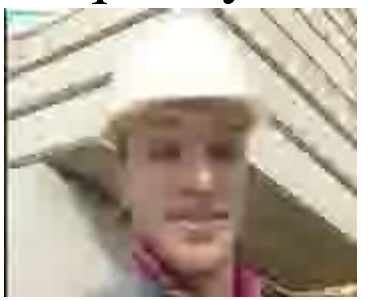
Temporal *vs* Quality transcoding (H.263)

	Temporal transcoder			Quality transcoder		
	Frames	PSNR1	Time(sec)	frames	PSNR1	Time(sec)
IR=256, 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	32.77	9.7
coastguard	118	32.39	9.5	300	32.47	9.3
IR=64, 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

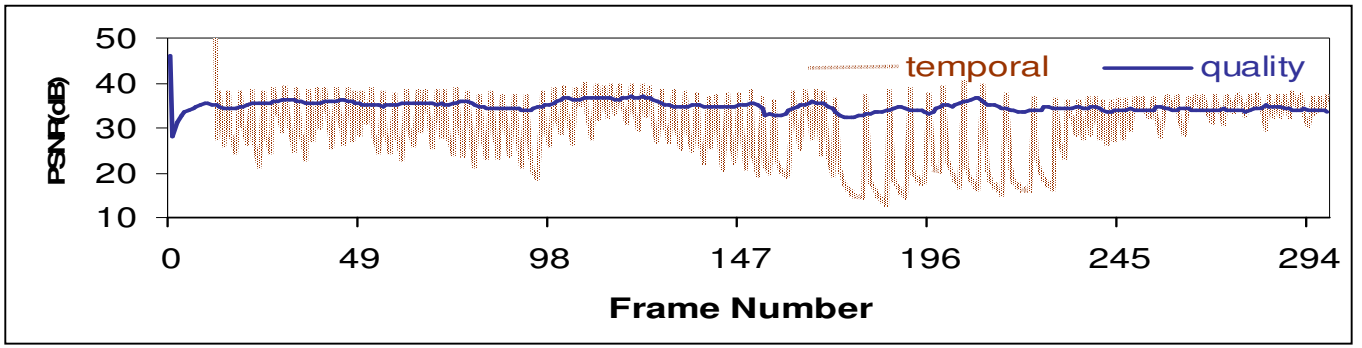
temporal



quality



Foreman video sequence, IR=256, R=128



H.264/ MPEG 4 Part 10: Advanced Video Coding

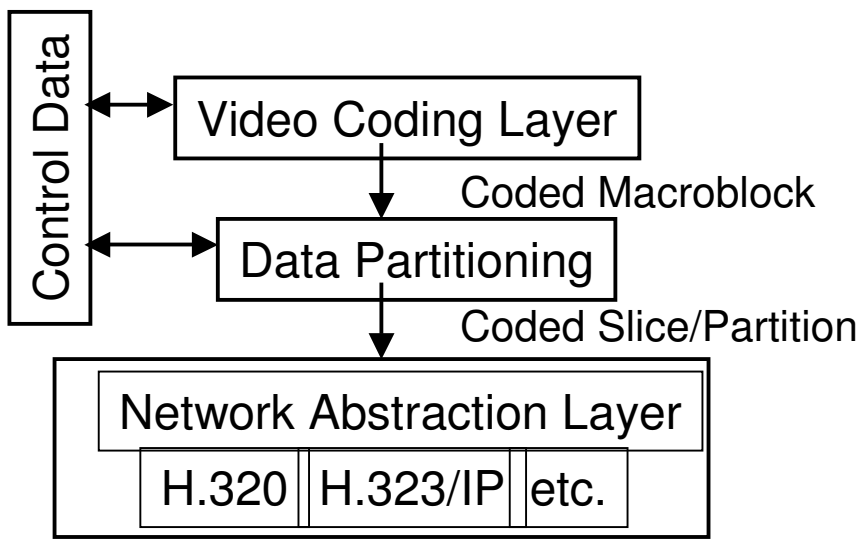
- In 1998, the Video Coding Experts Group (VCEG) issued a call for proposal on a project called H.26L, with the target to double the coding efficiency
- In 2001, VCEG and MPEG formed a Joint VideoTeam (JVT)
- In March 2003, the first draft of H.264 standard has been released

H.264 Scope

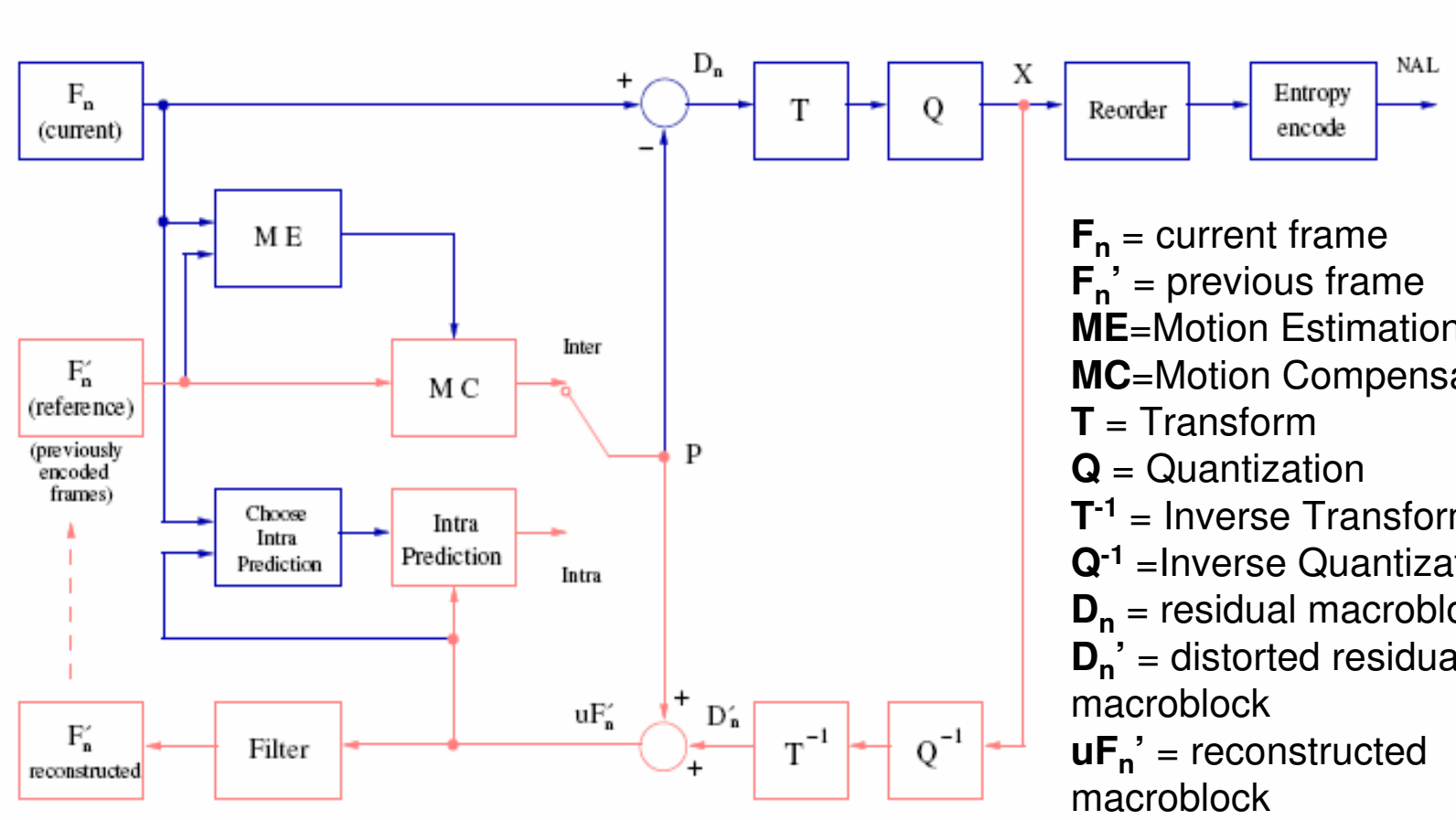
- It aims to achieve
  - Good quality at very low bit rate, by reaching very high compression rate
  - real-time
  - low end-to-end delay
- No great changes with respect to previous video coding standards, but a set of small improvements
- Many features, optional in previous coding standards, are mandatory in H.264 standard

H.264 Coding Layers

- Video Coding Layer:
  - designed to efficiently represent the video content
- Network Abstraction Layer:
  - formats the VCL representation of the video by adapting the coded bitstream to different transport protocols or storage media



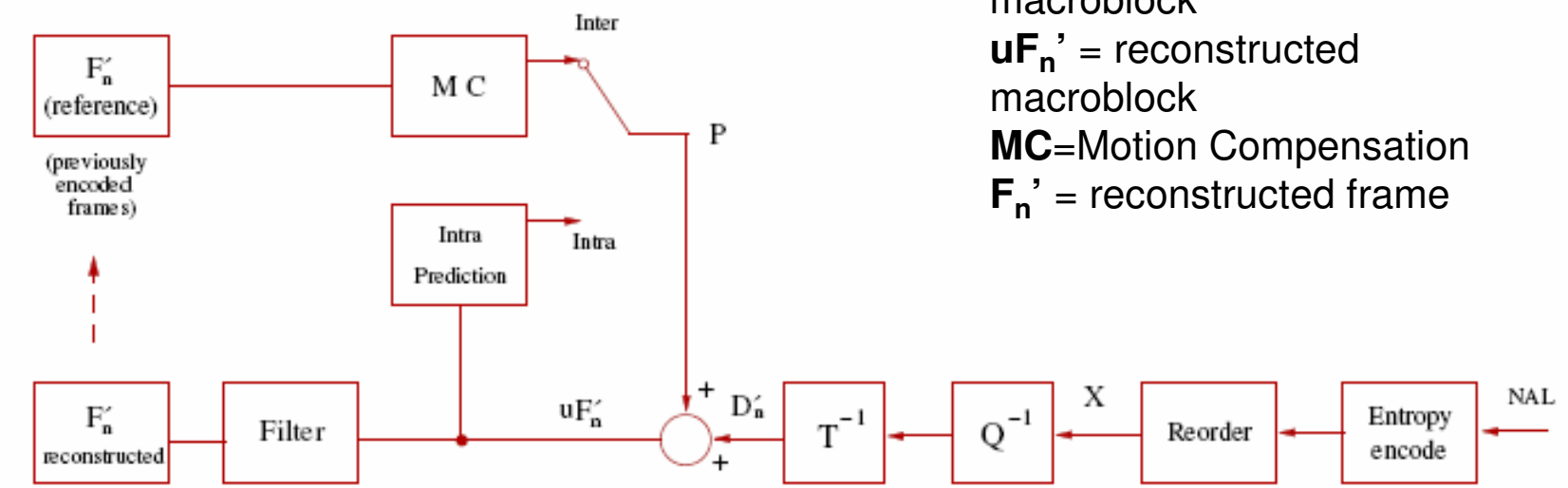
H.264/AVC Encoder



F_n = current frame
F_n' = previous frame
ME=Motion Estimation
MC=Motion Compensation
T = Transform
Q = Quantization
T^{-1} = Inverse Transform
Q^{-1} =Inverse Quantization
D_n = residual macroblock
D_n' = distorted residual macroblock
uF_n' = reconstructed macroblock
X = transformed+quantized block of coefficients

H.264/AVC Decoder

X = transformed+quantized block of coefficients
Q^{-1} =Inverse Quantization
T^{-1} = Inverse Transform
D_n' = distorted residual macroblock
uF_n' = reconstructed macroblock
MC=Motion Compensation
F_n' = reconstructed frame



H.264 improvements

- Variable and small block sizes (4×4)
- Quarter-pixel resolution
- Motion vectors beyond picture edges (optional in H.263)
- Moto compensation with multiple reference pictures
- Moto skipped inference
- Deblocking filter

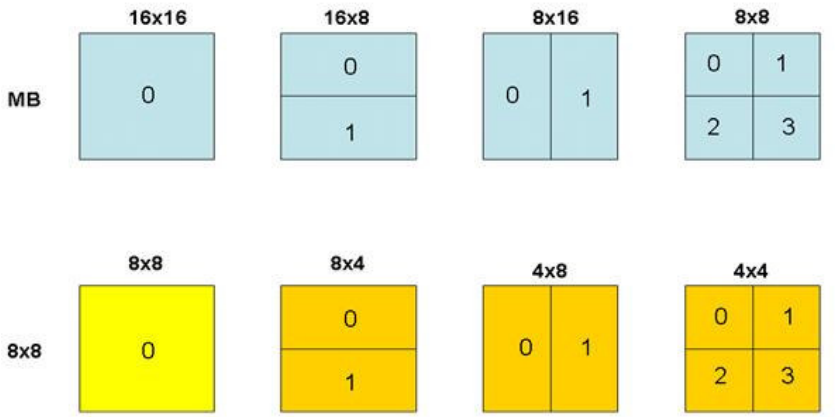
H.264 other improvements

- Entropic coding:
  - Transform on small size blocks
  - Transform with reduced word length (16 bit)
  - Arithmetic entropic coding
  - Adaptive entropic coding
- Robustness:
  - Flexible Slice/Macroblock Order
  - Redundant pictures
  - Data partitioning
  - Synchronization pictures (SI/SP)

H.264 coding

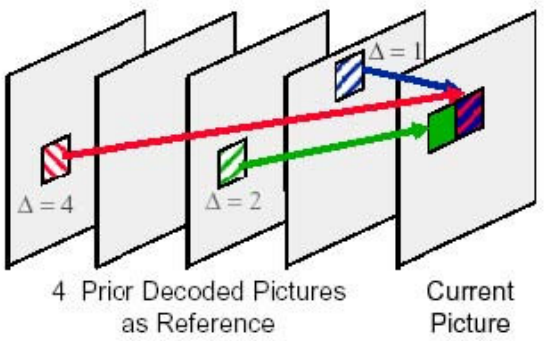
H.264 mean features:

- Variable and small block sizes (4×4)
- Quarter-pixel resolution
- NAL units
- New entropic coding algorithms (CABAC)
- Deblocking filter
- Moto compensation with multiple reference pictures



Average bit rate reduction!

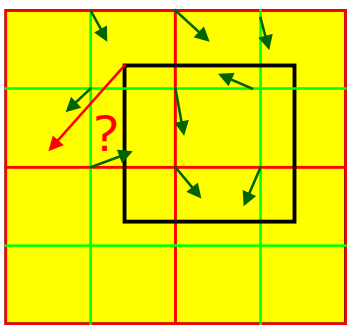
High complexity!



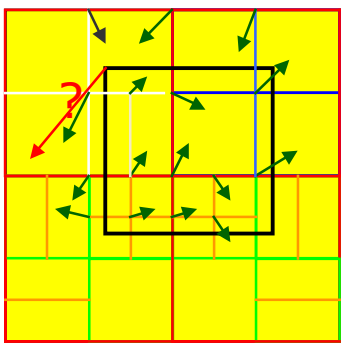
H.264 features

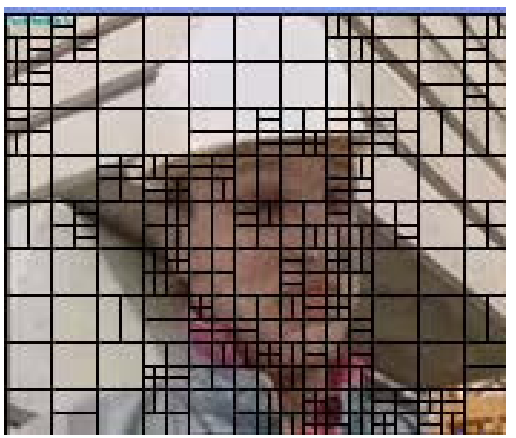
- **New** in H.264: variable macroblock partition (16 motion vectors for each macroblock)
- Transcoder keeps the same partitions of the remote encoder (most efficient solution)

Previous standards



H.264





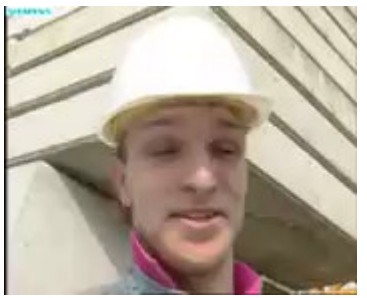
- How to apply MVC in H.264?
  - BI and TVC adaptation
  - **New MVC** algorithm

H.264 vs H.263 at high bit rate

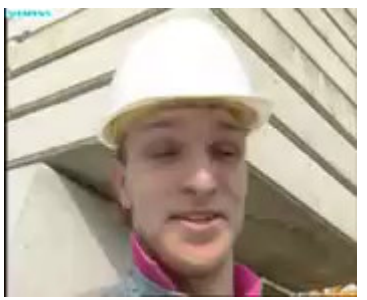
FOREMAN 128 kbps	# frames	Time (sec)	PSNR (dB)	PSNR2 (dB)
H.264 (16×16)	300	27 (5)	33.97	33.97
H.264 (16×16,16×8, 8×16,8×8)	300	31 (12)	34.84	34.84
H.264 (16×16,16×8, 8×16,8×8,8×4, 4×8, 4×4)	300	46 (26)	35.01	35.01
H.264 (all partitions + rate distortion optimiz.)	300	156 (23)	35.41	35.41
H.264 (all partitions + fast RD optimiz.)	300	137 (19)	35.40	35.40
H.264 (all partitions + 5 reference frames)	300	185 (150)	35.48	35.48
H.263 (all optionals enabled)	299	43	32.45	32.47

…and visually

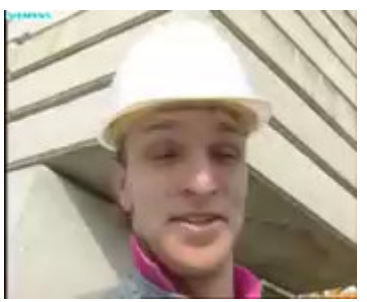
H.264 (16x16 only)



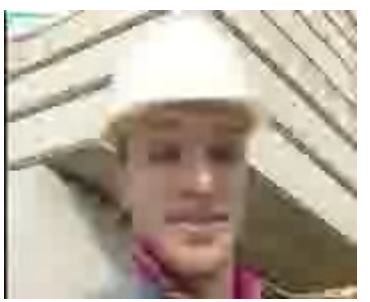
H.264 (16x16, 16×8, 8×16, 8×8, 4×8, 8×4, 4×4)



H.264 (all partitions+rd)



H.263

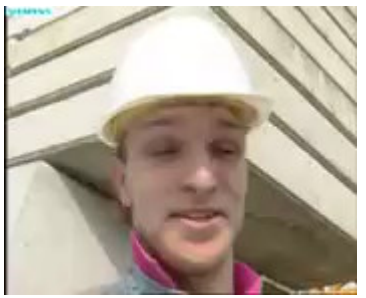


H.264 vs H.263 at low bit rate

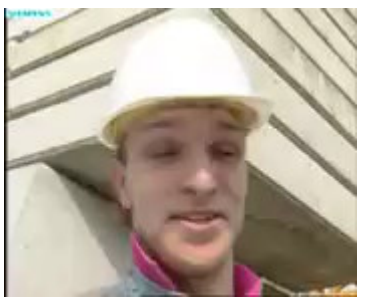
FOREMAN 32 kbps	#frames	Time (sec)	PSNR (dB)	PSNR2 (dB)
H.264 (16×16)	300	33 (8)	27.11	27.11
H.264 (16×16,16×8, 8×16,8×8)	300	35 (16)	27.65	27.65
H.264 (16×16,16×8, 8×16,8×8,8×4, 4×8, 4×4)	300	44 (27)	27.73	27.73
H.264 (all partitions + rate distortion optimiz.)	300	166 (29)	28.41	28.41
H.264 (all partitions + fast RD optimiz.)	300	116 (21)	28.39	28.39
H.264 (all partitions + 5 reference frames)	300	210 (177)	27.71	27.71
H.263 (all optionals enabled)	203	34	25.23	27.00

… but visually

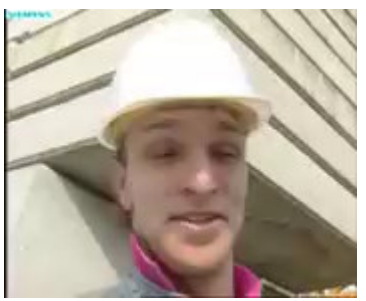
H.264 (16x16 only)



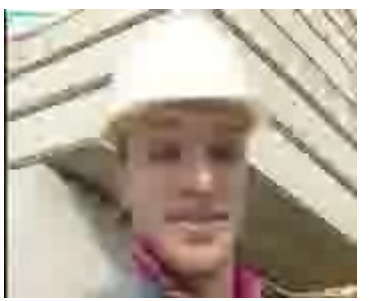
H.264 (16x16, 16×8, 8×16, 8×8, 4×8, 8×4, 4×4)



H.264 (all partitions+rd)



H.263



H.264 Profiles

- Baseline
  - Videotelephony
  - Videoconferencing
  - Wireless communications
- Main
  - Television broadcasting
  - Video storage
- Extended
  - Streaming media applications

H.264 Baseline Profile

- I and P picture types (not B)
- 1/4-sample motion compensation
- Tree-structured motion segmentation down to 4x4 block size
- Intra-prediction
- VLC-based entropy coding
- In-loop deblocking filter
- Flexible macroblock ordering/arbitrary slice ordering
- Some enhanced error resilience features

Optimizing H.264 encoder

- We operated some modifications to the reference software in order to obtain acceptable encoding times:
  - instead of computing all half and quarter pixels in two rounds, we compute them in only one round
  - fast way for choosing the optimal partitioning: instead of using the SAD (Sum of Absolute Differences) measure as decision parameter, we use other metrics:
    - the number of differences in terms of pixels
    - the maximum difference value
    - the average difference value
    - the most popular difference value

    compared with proper self-adjusting thresholds

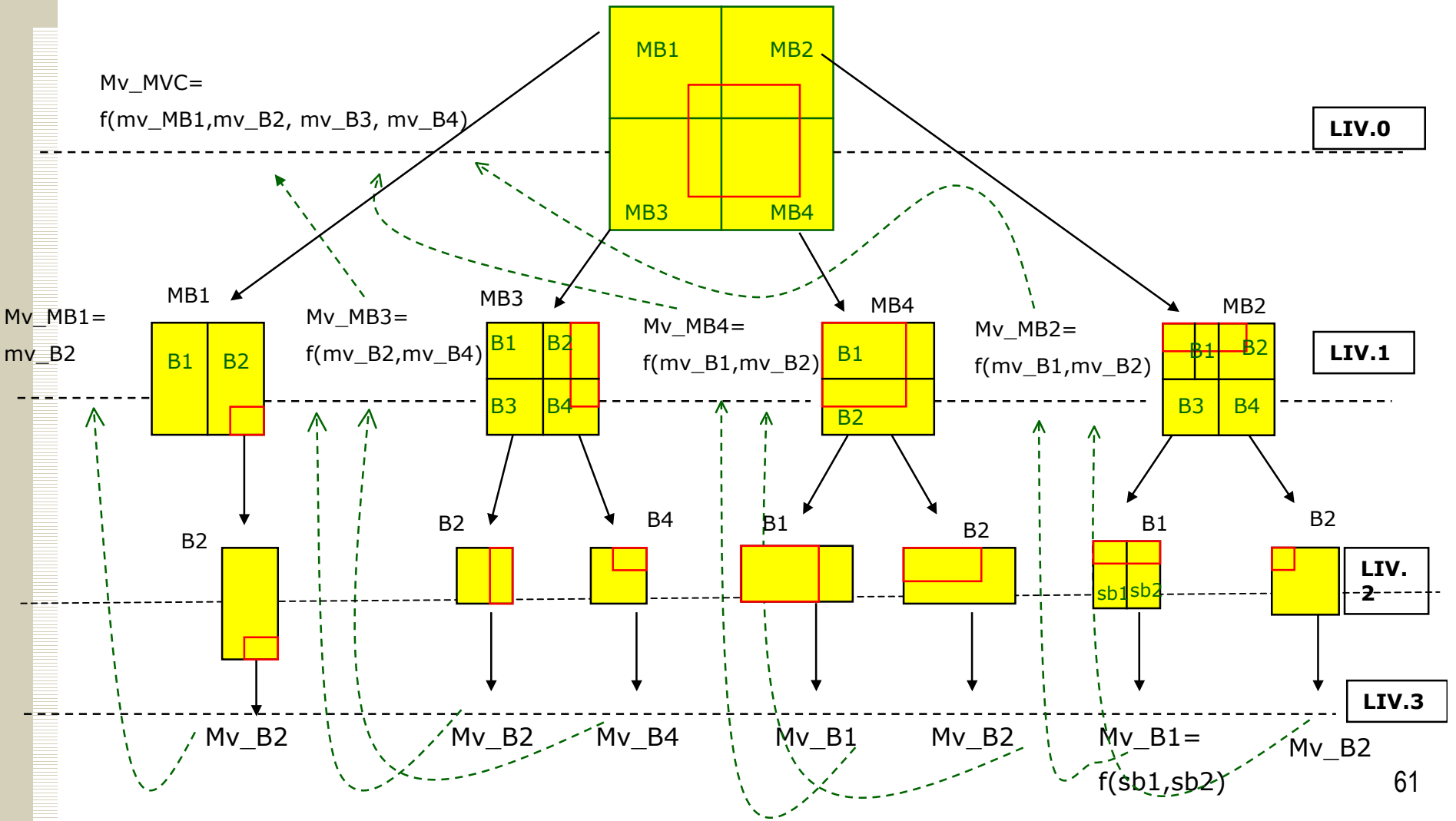
H.264 rate control

- Finally, we are implementing the TMN8 rate control algorithm to be used in the front encoder
- We think that, with a rate control algorithm able to skip frame in encoding phase, the transcoding process may be improved both in terms of quality and computation time

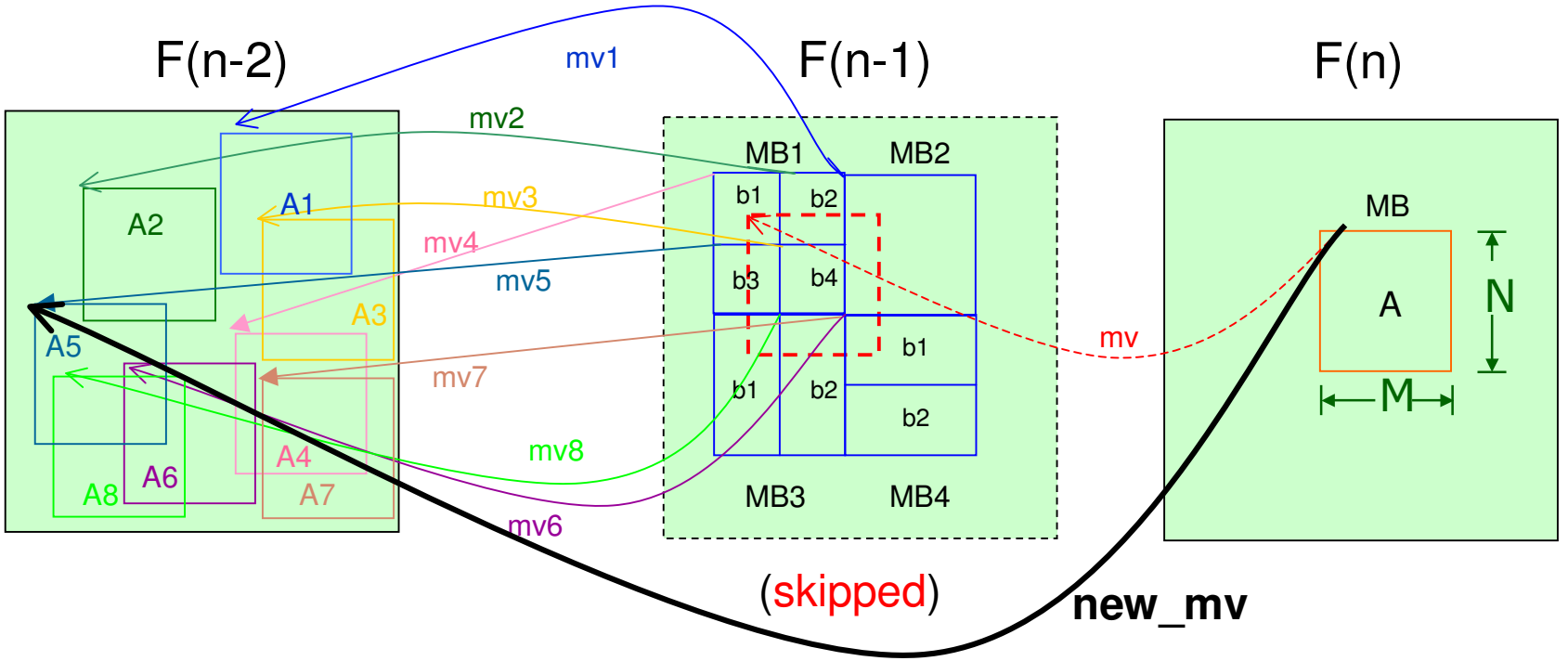
H.264 Transcoder

- We adopted the same architecture as we used for MPEG4/H.263 temporal transcoders
  - Motion Vector Composition (MVC)
  - Standard way of computing prediction errors
- Problems arose in MVC
  - Due to the variable partioning of frames, the motion vector composition is not trivial
  - We adapted 4 MVC algorithms present in literature to be used in variable partitioning
  - MVC follows the same tree-structured mechanism as in motion compensation

MVC in H.264

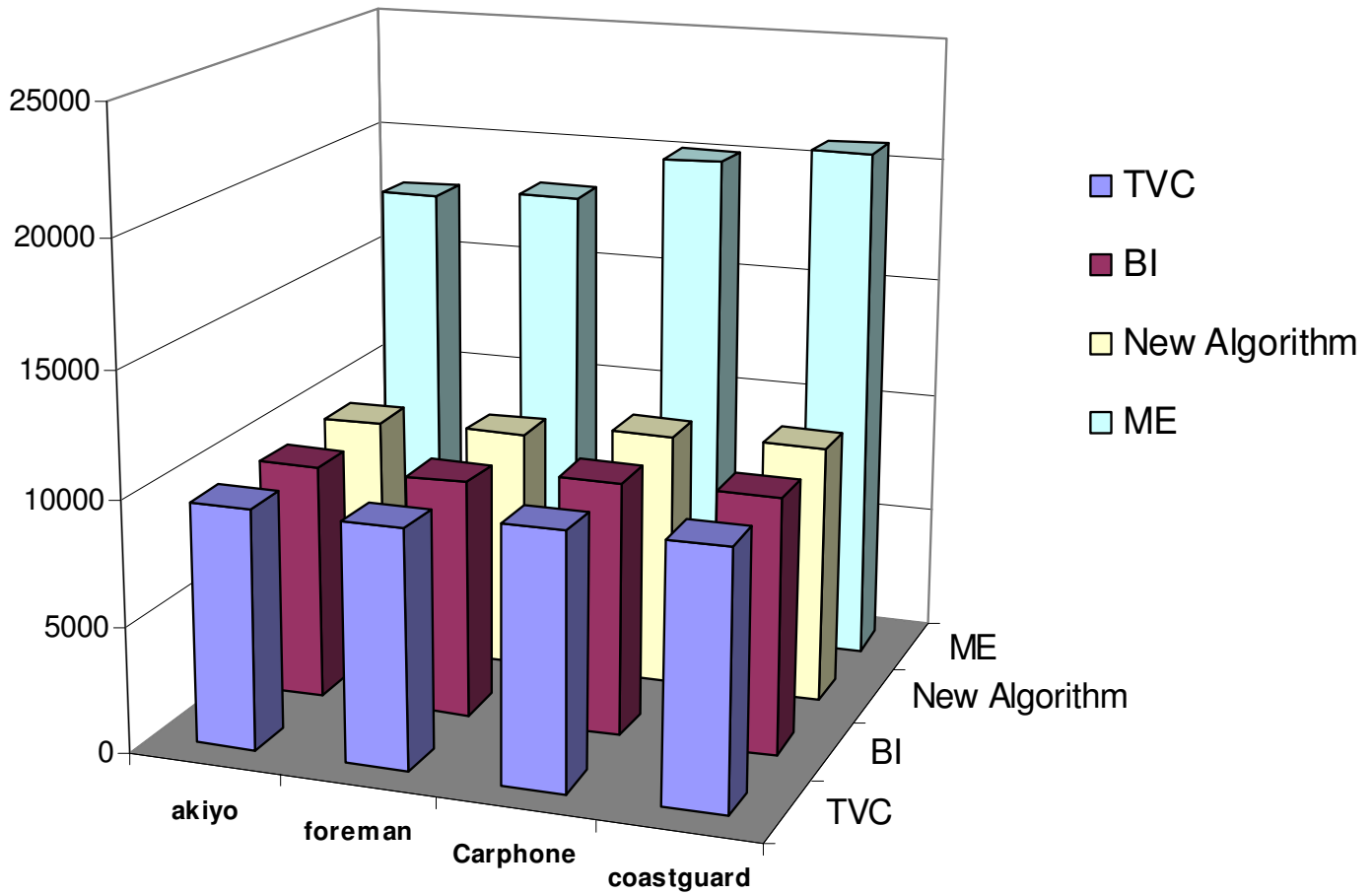

Mv_MVC=
f(mv_MB1,mv_B2, mv_B3, mv_B4)
MB1
MB2
MB3
MB4
LIV.0
Mv_MB1=
mv_B2
Mv_MB3=
f(mv_B2,mv_B4)
Mv_MB4=
f(mv_B1,mv_B2)
Mv_MB2=
f(mv_B1,mv_B2)
B1
B2
B3
B4
LIV.1
LIV.
2
LIV.3
sb1
sb2
Mv_B2
Mv_B2
Mv_B4
Mv_B1
Mv_B2
Mv_B1=
f(sb1,sb2)
Mv_B2

New MVC algorithm: example

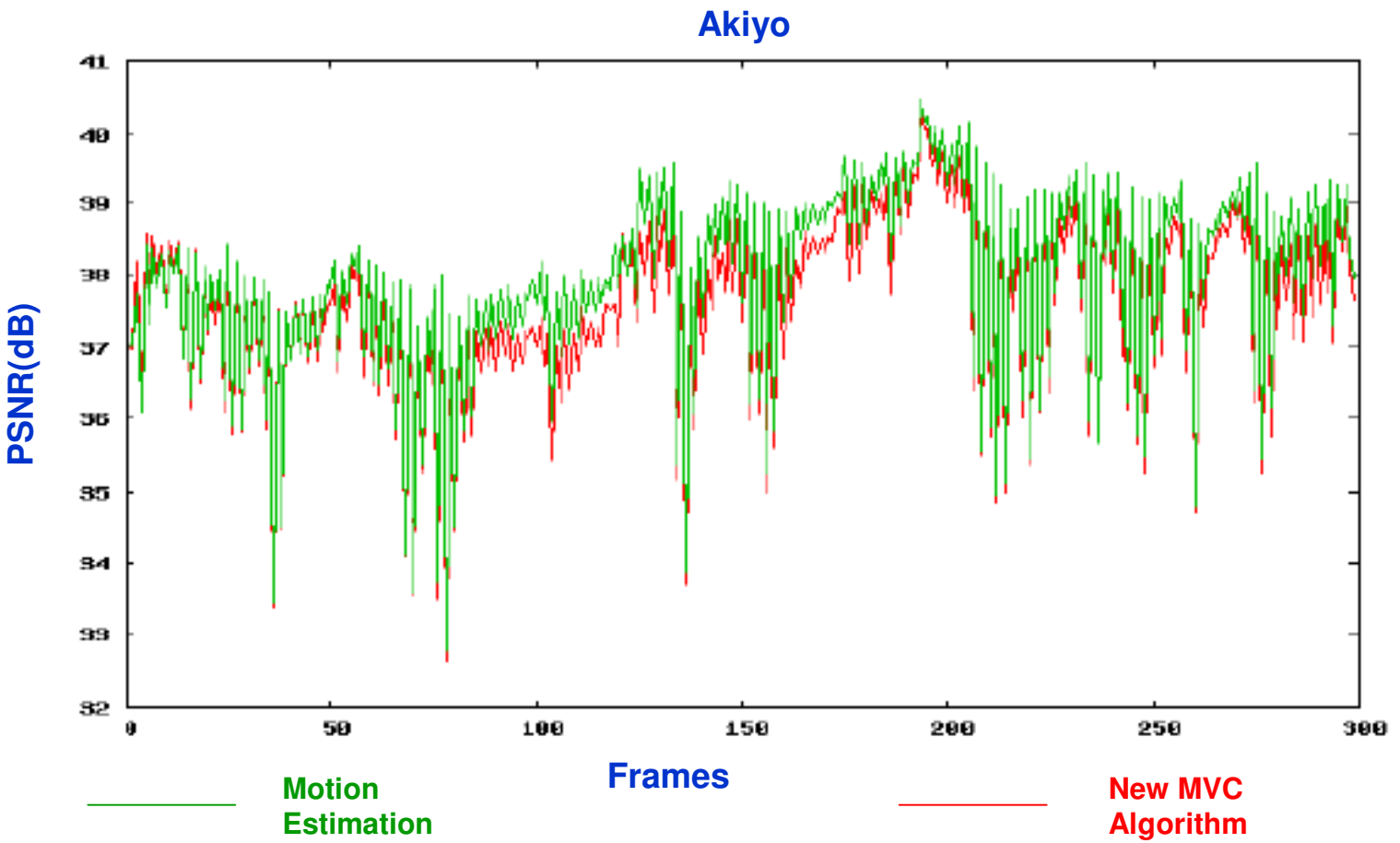


$$V^f = argmin_{i \in S} MSE(A, A_i) = argmin_{i \in S} (1/NxM \; |A-A_i|^2)$$

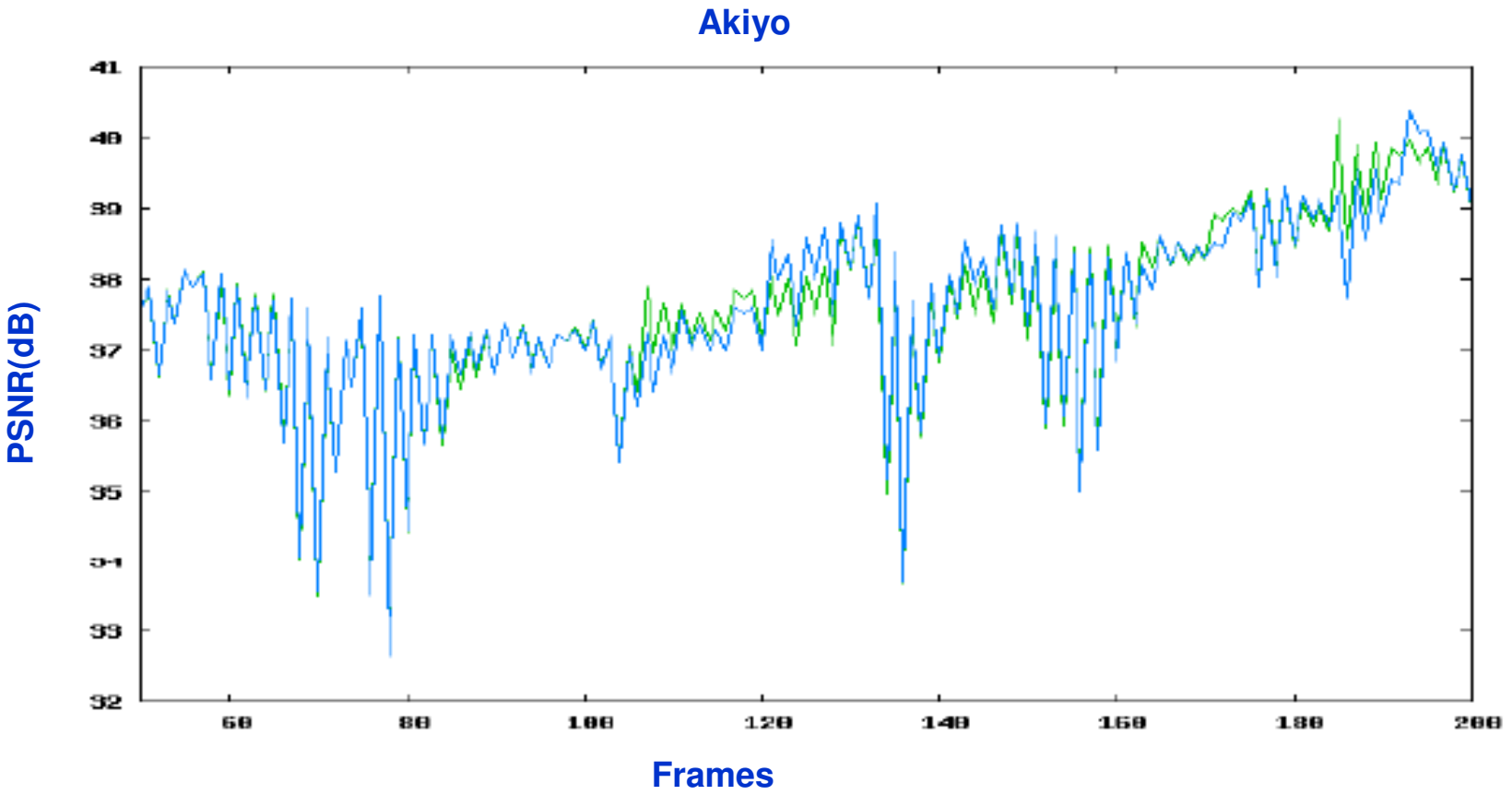
H.264 MVC evaluation



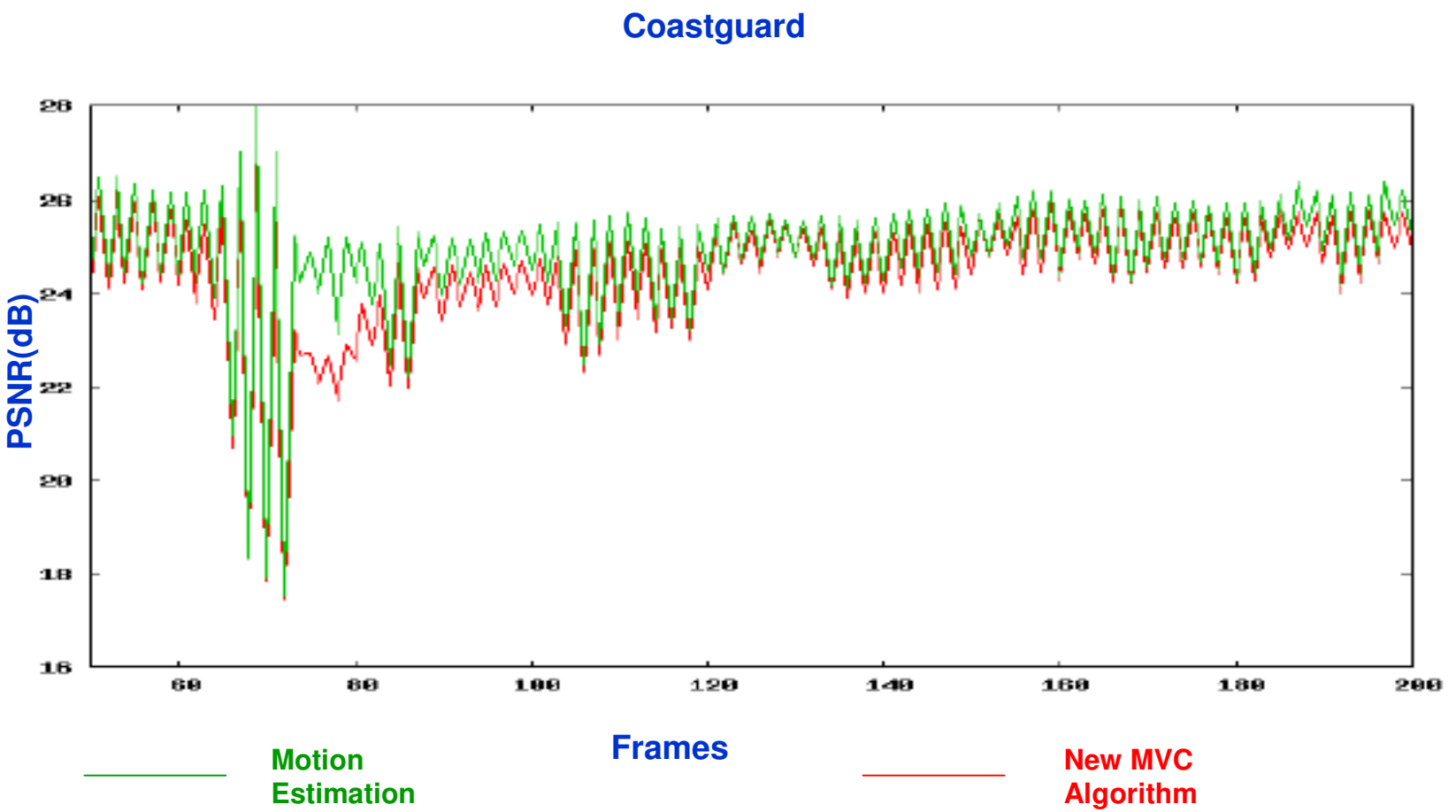
MVC performance



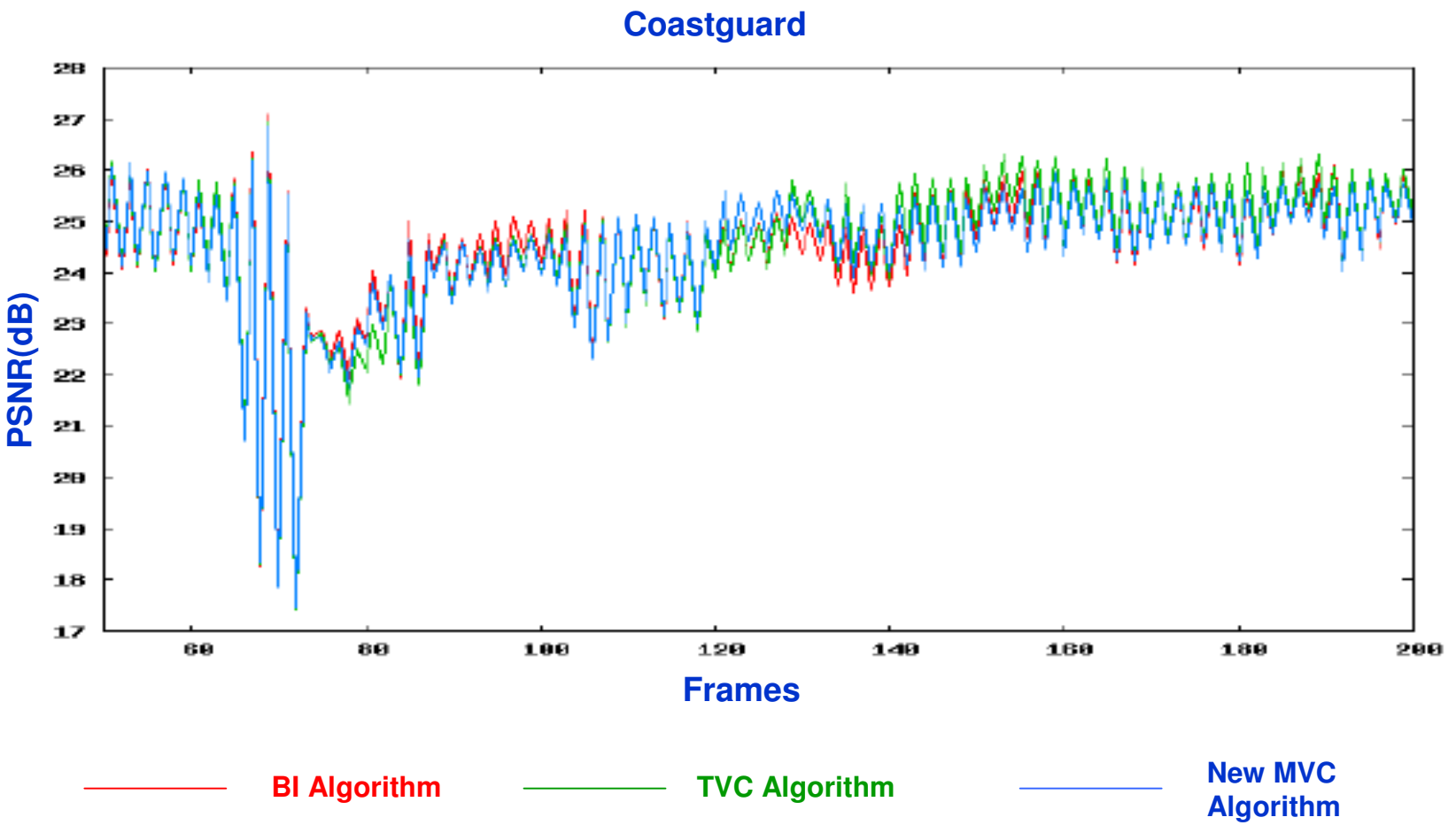
MVC performance



MVC performance



MVC performance



Conclusions

- We studied the video transcoding problem in real-time communications
- We developed temporal transcoders with MPEG4, H.263 and H.264 codecs
- We developed some skipping policies to be used in each transcoder
- We developed three MVC algorithms to be used in the H.264 transcoder

PISATEL

Future Work

- Develop an hybrid temporal/quality transcoding architecture
- Apply Frame Skipping Policies to H.264 Transcoder
- H.264 video transmission on MANET
- Error resilient features in transcoding

Pubblications

- M. A. Bonuccelli, F. Lonetti, F. Martelli. *Video Transcoding Architectures for Multimedia Real Time Services*, ERCIM News No. 62, pp. 39-40, July 2005.

- M. A. Bonuccelli, F. Lonetti, F. Martelli. *Temporal Transcoding for Mobile Video Communication*. In Proceedings of 2nd Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services (Mobiquitous 2005), pp.502-506, July 17-21, 2005, San Diego, CA.

- M. A. Bonuccelli, F. Lonetti, F. Martelli. *A Fast Skipping Policy for H.263 Video Transcoder*. In Proceedings of 12th International Workshop on Systems, Signals and Image Processing (IWSSIP'05). September 22-24, 2005, Chalkida, Greece.

Master Theses

- Luigi D'Amaro. *Algoritmi per la transcodifica video*.
- Gianni Rosa. *Transcodifica video per comunicazione mobile: studio di rate control*.
- Luca Leonardi. *Transcodifica video temporale: politiche di frame skipping*.
- Marina Paletta. *Realizzazione di un transcodificatore video temporale H.264 per video comunicazione mobile*.
- Riccardo Vagli. *Implementazione di un transcoder video basato sullo standard H.264/AVC*.
- Alsona Dema. *Rate Control in H.264*.

Acknowledgements

- We thank all ERI people who introduced us in this research area, for the helpful discussions and advices
- We thank PisaTel Lab people
- We thank all students who worked with us in this project