Università di Pisa

Dipartimento di Informatica
Scuola di Dottorato "GALILEO GALILEI"

Dottorato di Ricerca in Informatica

Ph.D. Thesis

Temporal Video Transcoding in Mobile Systems

Francesca Lonetti

Supervisor
Prof. M. A. Bonuccelli

October 15, 2007

To my mother

Acknowledgments

At the end of this work I want to thank all people sharing with me this experience. A great thank to Maurizio Bonucelli for his professional and friendly helpfulness.

I wish to thank Francesca Martelli, my special colleague and irreplaceable friend.

Thank to Antonia Bertolino who welcomes me in Pisatel Lab and to Ericsson Lab Italy for introducing me in this research area.

Thank to Eda, Andrea, Guglielmo, Sara, Jingua, Daniela, Antonino, Alberto, Cesare for their smile and suggestions.

Many thanks to my family for its love, in particular to my sister Pina.

Finally, a special thank to Enrico, my happiness and my support, I would never have arrived at this result without him.

Abstract

This thesis approaches the problem of temporal video transcoding in wireless networks. The general aim is to investigate temporal transcoding for improving real-time video communication of multimedia services by considering peculiar features of infrastructured and ad hoc wireless networks. We address the main issues of temporal transcoding, pointing out their characteristic problems and presenting our original proposals (motion vector composition algorithm and frame skipping policies). We focus on IEEE 802.11 vehicular networks for presenting a temporal transcoding system able to improve real-time video communication overcoming the typical network congestion and reducing late frames.

Contents

1 Introduction **1**
- 1.1 Thesis objectives 2
- 1.2 Organization of the thesis 5

I Background **7**

2 Video Coding **9**
- 2.1 Introduction 9
- 2.2 Video Compression 10
- 2.3 Video Codec 11
  - 2.3.1 Motion Estimation 13
  - 2.3.2 Motion Compensation 15
  - 2.3.3 DCT 16
  - 2.3.4 Quantization 17
  - 2.3.5 Entropy Coding 18
- 2.4 Video coding features 18
  - 2.4.1 Types of frames 18
  - 2.4.2 Rate Control 19
- 2.5 Video quality measure: PSNR 20

3 Video Transcoding **23**
- 3.1 Introduction 23
- 3.2 Why video transcoding 24
- 3.3 Video transcoding architectures 25
  - 3.3.1 Bit-rate reduction 26
  - 3.3.2 Spatial Resolution Reduction 30
- 3.4 Video Transcoding for Error-Resilience 32

4 Temporal Transcoding **35**
- 4.1 Introduction 35
- 4.2 Temporal Transcoding Architectures 36
  - 4.2.1 Pixel-domain temporal transcoder 37

4.2.2 DCT-domain temporal transcoder 39
4.3 Motion Vector Composition 41
4.3.1 Bilinear Interpolation 42
4.3.2 Forward Dominant Vector Selection 43
4.3.3 Telescopic Vector Composition 44
4.3.4 Activity Dominant Vector Selection 45
4.4 Motion Vector Refinement 45
4.5 Frame skipping policies 47
4.5.1 Motion-based frame skipping policies 47
4.5.2 Other frame skipping policies 49

II Proposed Approaches 53

5 Temporal Transcoding: architecture and Motion Vector Computation 55
5.1 Introduction 55
5.2 Transcoder Architecture 56
5.2.1 Transcoder architecture: reference frame not skipped 57
5.2.2 Transcoder architecture: reference frame skipped 57
5.3 MVC in our transcoder 59
5.4 MVC algorithms comparison 62
5.5 MVC in H.264 transcoding 63
5.5.1 Motivations 63
5.5.2 Multi-level MVC algorithm 70
5.5.3 Multi-Level Bilinear Scheme performance 74

6 Temporal Transcoding: skipping policies 79
6.1 Introduction 79
6.2 Communication model assumptions 81
6.2.1 Buffer-occupancy skipping policy 83
6.2.2 A New motion based skipping policy 84
6.2.3 Consecutive skipping policy 86
6.2.4 Random skipping policy 88
6.2.5 Size-prediction skipping policy 89
6.3 Performance analysis of skipping policies 92
6.3.1 Simulation Setting 92
6.3.2 Buffer-occupancy *vs* other skipping policies 94
6.3.3 New motion based *vs* Standard motion based skipping policy . 98

7 Temporal Transcoding in IEEE 802.11 Vehicular Networks 101
7.1 Introduction 101
7.2 IEEE 802.11 Distributed Coordination Function 103

7.3 Real-time video on IEEE 802.11 vehicular networks 105
7.4 Temporal transcoding on IEEE 802.11 vehicular networks 106
7.4.1 System assumptions 107
7.4.2 Our approach 108
7.4.3 System performance 110
7.4.3.1 Simulation settings 111
7.4.3.2 Simulation results 112

8 Conclusions 117

Bibliography 121

List of Figures

2.1 Video encoder block diagram 12
2.2 DPCM/DCT video encoder 13
2.3 DPCM/DCT video decoder 14
2.4 Typical MPEG4 video frame pattern 19

3.1 Universal Multimedia Access 25
3.2 Cascaded fully decoding/re-encoding scheme 26
3.3 Cascaded Pixel-Domain Transcoder (CPDT) 27
3.4 Open-loop transcoding architecture for bit rate reduction 27
3.5 Closed-loop transcoding architecture for bit rate reduction 28
3.6 Simplified DCT-domain transcoder (SDDT) 30
3.7 DCT-domain Motion Compensation 30
3.8 Spatial Resolution Reduction: Down-sampling of four motion vectors 31
3.9 Spatial Resolution Reduction: Down-sampling of four macroblock types 31

4.1 Cascaded Temporal Transcoder in the pixel-domain 37
4.2 DCT-domain temporal transcoder: residual signal re-computation for no-MC macroblocks 40
4.3 DCT-domain temporal transcoder: residual signal re-computation for MC macroblocks 40
4.4 Motion Vector Composition 41
4.5 Bilinear Interpolation algorithm 42
4.6 Forward Dominant Vector Selection algorithm 44
4.7 Telescopic Vector Composition algorithm 44
4.8 Activity Dominant Vector Selection algorithm 45
4.9 Fast Motion Vector Refinement algorithm 46
4.10 Variable step-size Motion Vector Refinement algorithm 47
4.11 Motion based frame skipping: motion change analysis (a) $n+1$ frame is not skipped (b) $n+1$ frame is skipped 49

5.1 Our temporal transcoder architecture 56
5.2 Our temporal transcoder architecture: not skipped reference frame . . 58
5.3 Our temporal transcoder architecture: skipped reference frame 58

5.4 MVC algorithms comparison: PSNR1 of “foreman” video sequence (from 30 to 15 fps) 68
5.5 MVC algorithms comparison: PSNR1 of “foreman” video sequence (from 30 to 7.5 fps) 68
5.6 MVC algorithms comparison: complexity ratio of “foreman” video sequence 69
5.7 Partition modes of a macroblock in H.264/AVC 70
5.8 H.264 temporal transcoding: motion vectors of partitions overlapping the reference area in the skipped frame 71
5.9 Multi-Level Bilinear Scheme (MLBS) 72
5.10 MLBS performance: PSNR1 of “foreman” video sequence (from 30 to 15 fps) 76
5.11 MLBS performance: PSNR1 of “foreman” video sequence (from 30 to 7.5 fps) 77
5.12 MLBS performance: complexity ratio of “foreman” video sequence . . 77

6.1 Skipping policies: communication model 83
6.2 Temporal transcoding: size of video frame after skipping a number of previous consecutive frames 90
6.3 Buffer-occupancy *vs* New motion based skipping policies: PSNR1 of “coastguard” video sequence, *IR*=64 Kbps, *R*=32 Kbps 97
6.4 Buffer-occupancy *vs* Consecutive skipping policies: PSNR1 of “foreman” video sequence, *IR*=128 Kbps, *R*=32 Kbps 97
6.5 Buffer-occupancy *vs* Size-prediction skipping policies: Complexity ratio of “foreman” video sequence 98
6.6 New motion based *vs* Standard motion based skipping policies: PSNR1 of “mobile” video sequence, *IR*=256 Kbps, *R*=128 Kbps 99
6.7 New motion based *vs* Standard motion based skipping policies: transcoded frames (210-213) of “table tennis” video sequence, QCIF format, *IR*=128 Kbps, *R*=32 Kbps 99

7.1 IEEE 802.11 Distributed Coordination Function: RTS/CTS Access Mechanism 104
7.2 IEEE 802.11 wireless network: communication model 107
7.3 IEEE 802.11 wireless network: decoding process at the receiver side . 108
7.4 Vehicular network: simulated road scenario 111
7.5 Proposed temporal transcoding system on IEEE 802.11 protocol *vs* Conventional IEEE 802.11 protocol: PSNR values of “foreman” video sequence N0→N2, I experiment. 114
7.6 Proposed temporal transcoding system on IEEE 802.11 protocol *vs* Conventional IEEE 802.11 protocol: PSNR values of “foreman” video sequence N2→N0, I experiment. 115

List of Tables

2.1 Video frame formats 11

5.1 Pseudo-code of MVC module 61
5.2 MVC algorithms performance comparison: PSNR1 of different video sequences at different input bit rate IR(Kbps) from 30 to 15 fps . . . 64
5.3 MVC algorithms performance comparison: PSNR1 of different video sequences at different input bit rate IR(Kbps) from 30 to 7.5 fps . . . 65
5.4 MVC algorithms performance comparison: Complexity ratio of different video sequences at different input bit rate IR(Kbps) from 30 to 15 fps 66
5.5 MVC algorithms performance comparison: Complexity ratio of different video sequences at different input bit rate IR(Kbps) from 30 to 7.5 fps 67
5.6 MLBS algorithm performance: PSNR1 of different video sequences at different input bit rate IR(Kbps) from 30 to 15 fps 75
5.7 MLBS algorithm performance: Complexity ratio of different video sequences at different input bit rate IR(Kbps) from 30 to 15 fps . . . 76

6.1 Pseudo-code of Buffer-occupancy skipping policy 84
6.2 Pseudo-code of New motion based skipping policy 87
6.3 Pseudo-code of Consecutive skipping policy 88
6.4 Pseudo-code of Random skipping policy 89
6.5 Pseudo-code of Size-prediction skipping policy 91
6.6 Buffer-occupancy skipping policy: TF and PSNR1(dB) of "foreman" video sequence for different IR(Kbps), R(Kbps) and τ values 94
6.7 Proposed skipping policies: TF, PSNR1(dB) and PSNR2(dB) for different video sequences, IR(Kbps) and R(Kbps) values 95
6.8 Proposed skipping policies: TF, PSNR1(dB) and PSNR2(dB) for other video sequences, IR(Kbps) and R(Kbps) values 96
6.9 New motion based *vs* Standard motion based skipping policies: TF, PSNR1(dB) and PSNR2(dB) for different video sequences, IR(Kbps) and R(Kbps) values 100

7.1 Pseudo-code of temporal transcoding system on IEEE 802.11 110

7.2 QP and PSNR of different video sequences after coding and decoding 112
7.3 Conventional IEEE 802.11 protocol: PSNR *vs* required bandwidth of video flows with network congestion (I experiment) 113
7.4 Conventional IEEE 802.11 protocol: PSNR *vs* required bandwidth of video flows with network congestion (II experiment) 114
7.5 Proposed temporal transcoding system on IEEE 802.11 protocol: PSNR *vs* required bandwidth of video flows (I experiment) 114
7.6 Proposed temporal transcoding system on IEEE 802.11 protocol: PSNR *vs* required bandwidth of video flows (II experiment) 115

Chapter 1

Introduction

Abstract

In this chapter we introduce the topic of our research, the problem we faced, and the original proposed solutions.

Advanced types of interactive and multimedia services, such as Digital TV broadcasting, Distance Learning, Video on Demand, Video Telephony and multipoint Video Conferencing are used in everyday life, for working or practical purposes. The great development of these services in the last years is due to the following two important factors:

- improved digital video technologies in compression (encoding) and decompression (decoding), that allow high reduction of bandwidth intensive digital video to a manageable size for transmission or storage. Highly efficient and scalable video coding standards have been proposed for various applications, such as H.263 for low-bit rate video communications, MPEG2 for broadcasting and general high quality video application, MPEG4 for streaming video and interactive multimedia applications, H.264 for high compression requests;
- advances in network topology deployment, allowing a rapid and easy access to media content. Large diffusion of infrastructured (cellular networks, UMTS) and ad hoc networks (Wi-Fi, vehicular networks) offers to users mobility, flexibility and mobile access to internet.

The quality of multimedia services is influenced by devices capabilities on communication, processing, storage and display. Adapting *on-the-fly* the media content to different device and network characteristics (channel bandwidth and terminal complexity, for instance), is one of the most important problems in this setting. A typical strategy to approach this problem is the content adaptation, better known as transcoding, usually performed by servers of a communication system, or by gateways interconnecting heterogeneous networks.

Multimedia comprises images, video, audio, graphics, text, and the so called presentation of media content. In this thesis, we focus on delivering of video content. Video transcoding is defined, in general, as the conversion of one compressed video stream to another one with different features, without performing the total decoding and re-encoding process. To enable interoperability among devices with different bandwith constraints and computational capacities, different kinds of transcoding are dinamically required, depending on network resources and device features.

Video transcoding can provide format conversion, resolution scaling (spatial transcoding), bit rate conversion (quality transcoding), frame rate conversion (temporal transcoding). Format conversion operates a syntax change from a video coding standard into another one. Spatial transcoding, reduces the spatial resolution of a compressed video. It is required for facing the problem of limited size in many access terminals. Quality transcoding operates on the bit allocation for each frame, by tuning the quantization parameters, according to the target bit rate. Temporal transcoding is a process that skips some frames in order to change the frame rate of the video sequence, without decreasing the video quality of not skipped frames. A critical issue in the design of multimedia applications is to guarantee real-time constraints for enabling a good service quality.

In this thesis we are interested in temporal transcoding. Our goal is to investigate temporal transcoding issues for improving real-time video communication of multimedia services (such as video telephony, video conference, etc), in mobile networks, more specifically infrastructured and ad hoc networks.

We outline the objectives and contributions of the thesis in Section 1.1, presenting its organization in Section 1.2.

1.1 Thesis objectives

We investigate the main issues of temporal transcoding for improving real-time video communication. For this purpose, the proposed solutions try to reduce the time needed to perform the transcoding process and to achieve a good video quality by considering peculiar features of different network topologies. More specifically, we focus on temporal transcoding for infrastructured and vehicular ad hoc networks.

First of all, after describing the features of existing transcoding architectures in Chapters 3 and 4, we design a pixel-domain temporal transcoding architecture, and we address the main problem of motion vector computation when a frame is skipped. Motion Vector Composition (MVC) is a well know technique adopted to greatly reduce computation time of heavy motion estimation processes. We evaluate the performance of the most popular motion vectors composition algorithms (Bilinear Interpolation [HWL98], Forward Dominant Vector Selection [YSL99], Telescopic Vector Composition [SG00] and Activity Dominant Vector Selection [CCP02]), in our architecture design.

Then, we address the problem of motion vector computation in H.264 temporal transcoding, and propose a MVC algorithm for H.264 based transcoding. An important feature of the H.264 codec, which is of interest in this thesis, is the variable block-size partitioning of frames, as described in Section 5.5.1. Due to this variable block-size partitioning, the existing MVC algorithms ([HWL98][YSL99][SG00] [CCP02]) cannot be applied without changes, since a separate motion vector has to be considered for each partition or sub-partition, overlapping the reference area pointed by the motion vector of the current macroblock in the skipped frame. To overcome this problem, we propose a multi-level motion vector composition scheme, together with the Bilinear Interpolation function, that takes into account the variable block size partitioning of H.264 coded frames.

Since many multimedia services are not specifically tailored to mobile systems, often the channel bandwidth required for transmission does not match application needs. In mobile systems, with an infra-structured network, transcoding is needed at interworking nodes to perform bit rate reduction, or more generally to deal with the current heterogeneous communication infrastructure and the diversity of services and user terminals.

Many proposed solutions perform variable and constant bit rate reductions by applying frame rate control schemes aiming to tune the quality of transcoded frames minimizing distortion in the reconstructed signals.

Temporal transcoding performs this bit rate reduction by skipping frames only, so it allows a good quality of transcoded frames when a high bit rate reduction is needed also, with a consequent jerky effect due to skipping of frames.

In this thesis, we investigate several strategies for choosing frames to be skipped (called frame skipping policies) in order to minimize this jerky effect, when a constant bit rate reduction is in order. In addition, to guarantee real-time constraint of many multimedia services, we outline frame skipping issues having a minimum processing delay and a communication delay compliant with the time requirements of these real-time services.

Most popular frames skipping policies ([HWL98][FCS02] [SC04]) base the choice of frames to be skipped on motion information. The aim is to reduce the jerky effect of the displayed sequence. The basic idea of this approach is that a frame with a lot of motion has not to be skipped, since it is not possible to replace it with the previous frame at displaying time, without introducing considerable quality degradation.

However, because of new challenges introduced in video communications (i.e., real-time constraints, high bit rate reduction in mobile systems), buffer control is, in our opinion, the most important factor to have a good quality of transcoded video sequences in real-time services. So, we investigate buffer constraints in frame skipping problem. We describe our communication model assumptions, and we propose a basic buffer based skipping policy able to deal with real-time requirements of this communication model. In addition, we propose the following other skipping policies that improve the quality of transcoded video sequence when buffer constraints are met:

- a new motion based skipping policy considering the different types of motion in a frame;
- a policy that minimizes the number of consecutive skipped frames;
- a policy using a randomized approach related to buffer constraints.

Finally, we propose a frame skipping policy also able to reduce the processing delay of the transcoding system, by predicting the size of the next frames when the current one is skipped, according to a logarithm function.

After designing the above frame skipping policies dealing with real time constraints in constant bit rate infrastructured networks, we investigate the temporal transcoding issues in other network system, such as ad hoc networks.

Another original contribution of this thesis in the field is approaching temporal transcoding for improving real-time video communication in IEEE 802.11 vehicular ad hoc networks. Inter-vehicular wireless networks are recently gaining much attention in the research community, due to the number of applications that could improve the quality of everyday life. Real time video communication is required for enhancement of road safety, by propagating emergency alerts or personal and entertainment applications.

The transmission of real time video sequences in vehicular ad hoc networks incurs in some problems, in particular due to transmission errors, variable bandwidth and channel access delay, that bring to packet loss and variable delay of video delivering.

The known solutions mainly aim to develop new access protocols dealing with channel errors and packet delay. Our approach adopts currently available wireless networking protocols, such as the widely used IEEE 802.11 Wireless Local Area Network standard, without any changes. This makes a large and not expensive application of our system suitable.

We investigate the behaviour of basic Distributed Coordination Function (DCF) of IEEE 802.11 protocol, focusing on the access channel delay when network load is moderate or heavy, so congestion can occur. For real-time multimedia applications, this access channel delay causes performance degradation, since a frame received after the deadline is not displayed. This has two effects: the former is wasting bandwidth; the latter is delaying the successive frames, which further degrades the performance of real-time applications.

We develop a temporal transcoding system that discards frames when they are late. We assume a cross-layer technique allowing interaction between Medium Access Control (MAC) and application layers. Note that, cross layer technique is not the focus of our work. Using temporal transcoding, our solution aims to overcome congestion reducing bandwidth needs, and decreases delayed packets delivery.

The technique proposed in this thesis is about real-time video transmission in vehicular networks. However, our approach can be extended to consider other types of ad hoc networks.

1.2 Organization of the thesis

The thesis is organized substantially in two parts: the former (Chapters 2, 3, 4) reviews the existing solutions for video transcoding focusing on temporal ones; the latter (Chapters 5, 6, 7) reports the original results of our work.

In particular, Chapter 2 describes the video coding features concerning the transcoding process.

In Chapter 3, after outlining goal and issues of transcoding, we briefly survey existing architectural solutions to perform bit rate and spatial resolution reductions.

Chapter 4 is devoted to overview the most important issues in temporal transcoding: architecture design, motion vector computation, and frame skipping policies. We outline the advantages of DCT and pixel domain transcoding architectures for the frame skipping problem. We address motion vector composition technique and we describe the existing motion vector composition algorithms. Finally, the known frame skipping strategies are presented. This temporal transcoding overview has been presented in [LM07b].

In Chapter 5, we describe our temporal transcoding architecture. We evaluate performance of the motion vector composition algorithms presented in Chapter 4, and we address the problem of Motion Vector Composition in an H.264 based transcoder. We propose a multi level motion vector composition algorithm dealing with variable partitioning of H.264 coded frames. This algorithm has been published in [LM07a].

In Chapter 6, we face the frame skipping issues, to deal with real-time requirements of multimedia applications in infrastructured networks with constant bit rate reduction. We propose several frame skipping strategies. They are all based on output buffer constraints and provide other metrics to reduce the computational delay of transcoding process and improve the quality of the transcoded video sequence. These results have been published in [BLM05a] [BLM05b] [LM06].

Chapter 7 addresses the temporal transcoding for real time video communication in a recently developed type of network: Vehicular Ad Hoc Networks (VANET). We use IEEE 802.11 available technology to design a system that, by skipping late frames, is able to improve the quality of a real time video, while reducing the bandwidth consumption. This approach has been presented in [BGLM07].

Finally, in Chapter 8, conclusions and topics for future studies are highlighted.

Part I

Background

Chapter 2

Video Coding

Abstract

In this chapter, we describe the main concepts of the video coding process. We present a high level overview of the basic structure of a video codec, focusing on the coding features concerning the transcoding process.

2.1 Introduction

Pervasive and high-quality digital video has been the goal of companies, researches and standard bodies over the last two decades. Recent development of multimedia services and applications is due to improved digital video technologies. Highly efficient and scalable video compression formats enable many advanced types of interactive and distribution services, such as Digital TV broadcasting, Distance Learning, Video on Demand, Video Telephony and multipoint Video Conferencing. Digital video is an increasing technology which will continue to pervade business, networks and homes.

Getting digital video from its source (a camera or a stored clip) to its destination (a display) involves a chain of processes and components. The key processes in this chain are compression (encoding) and decompression (decoding), in which bandwidth intensive digital video is first reduced to a manageable size for transmission or storage, and then reconstructed for display.

Even with constant advances in storage and transmission capacity, compression is an essential component of multimedia services. Video compression makes possible to use digital video in transmission and storage environments that would not support uncompressed video. For example, current Internet throughput rates are not sufficient to handle uncompressed video in real-time (even at low frame rates and/or small frame size). Digital Versatile Disk (DVD) video storage would not be practical without video and audio compression. In addition, video compression enables a more efficient use of transmission and storage resources. If a high bit rate transmission channel is available, video compression allows to send high-resolution compressed video or multiple compressed video streams instead of sending a single, low-resolution, uncompressed stream. This brings a keen interest in the continuing development and improvement of video compression and decompression methods and systems. The objective is to provide a better image quality, more reliable and flexible solutions.

In the last years, many video coding standards have been proposed for various applications, such as H.263 for low-bit rate video communications, MPEG1 for storage media applications, MPEG2 for broadcasting and general high quality video application, MPEG4 for streaming video and interactive multimedia applications, H.264 for high compression requests. Each standard primarily defines a coded representation (or syntax) that describes visual data in a compressed form and a method of decoding the syntax to reconstruct visual information. The standards specifically do not define an encoder; rather, they define the output that an encoder should produce. The standard aims to ensure that compliant encoders and decoders can successfully inter-work, while allowing to developers the freedom to produce competitive and innovative products.

In this chapter, we describe the basic concepts of the video coding process that are common to main video coding standards, and which are useful for understanding video transcoding issues.

The experimental results presented in this work are mainly obtained by using MPEG4 and H.263 video coding standards and MPEG4 and H.263 based transcoder implementations. Particular features of H.264 (the last video coding standard) are taken into account to propose our solutions for improving the transcoding process.

2.2 Video Compression

Video coding is the process of compressing and decompressing a digital video signal. Compression involves a complementary pair of systems, the encoder and the decoder. The encoder converts the source data into a compressed form (using a reduced number of bits) for practical transmission or storage, and the decoder converts the compressed form back into a pixel-based representation of the digital video data. The encoder/decoder pair is often indicated as a *CODEC*.

Digital video is a representation of a real-world visual scene sampled spatially and temporally. Sampling is repeated at regular intervals in time (e.g. 1/25 or 1/30 second intervals) to produce a moving video signal. The sampling unit is called frame, or picture, or image. Each spatio-temporal sample (picture element or pixel) is represented by a number or set of numbers describing the brightness (luminance) and colour of the sample. In RGB (Red, Green and Blue) colour space, a colour image sample is represented with three 8 bits numbers, indicating the relative proportions of Red, Green and Blue. A more effective and popular way of efficiently representing colour images is YCbCr (sometimes referred to as YUV).

Table 2.1: Video frame formats

Format	Luminance resolution (horiz. x vert.)
Sub-QCIF	128×96
Quarter CIF (QCIF)	176×144
CIF	352×288
4CIF	704×576
16CIF	1408×1152

It separates the luminance signal (Y) from the colour information given by three colour components (chrominance) Cb, Cr, Cg, representing the difference between the blue, red and green colour intensity and the luminance of the image sample. In the YCbCr colour space, only the luma (Y) and blue and red chrominance (Cb, Cr) need to be stored or transmitted, since the third colour component can always be computed from the other two. One of the advantages of this format is that Cb and Cr components may be represented with a lower resolution than luminance, without affecting the perceived video quality, since the human eye is less sensitive to colour variations than to luminance variations.

There are different frame resolutions. The Common Intermediate Format (CIF) is the basic one for a popular set of formats listed in Table 2.1. The choice of frame resolution depends on the application and on the available storage or transmission capacity. For example, 4CIF is appropriate for standard-definition television and DVD-video; CIF and QCIF are popular for videoconferencing applications; QCIF and SQCIF are appropriate for mobile multimedia applications where the display resolution and the bit rate are limited [I.E03].

There is high correlation between temporally adjacent frames and between pixels that are close to each other in the same frame. Video compression is achieved by removing this temporal and spatial redundancy respectively. The major video coding standards, as H.261, H.263, MPEG1, MPEG2, MPEG4 and H.264, are based on the same "codec model" that we shall explain in the following.

2.3 Video Codec

The main purpose of a video codec is to achieve compression efficiency with high video quality. These two goals are usually conflicting, because a lower compressed bit rate typically produces worse image quality at the decoder.

A video encoder, as shown in Figure 2.1, consists of three main functional units: a temporal compression model, a spatial compression model, and an entropy compression unit.

The temporal compression model reduces the similarities between neighbouring video frames by performing a prediction process of the current frame with respect

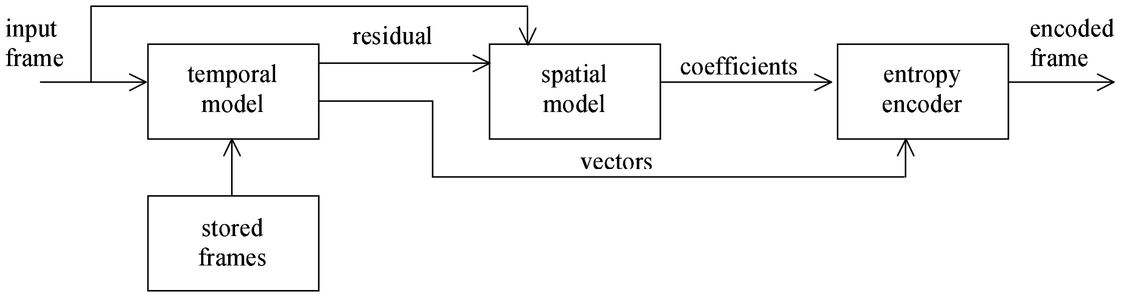


Figure 2.1: Video encoder block diagram

to one or more previous or future frames (reference frame). The result is a set of motion vectors, and a residual frame (created by subtracting the reference from the current frame). The spatial compression model reduces spatial redundancy between neighbouring samples in residual frame by applying a transformation into another domain (DCT is the most popular transformation), and quantizing the results (quantization) to remove insignificant values. The output is a set of quantized transform coefficients. These coefficients, together with the motion vectors, are compressed by the entropy encoder to remove statistical redundancy (commonly-occurring vectors and coefficients are represented by short binary codes). Note that, in some standard codecs, the spatial compression model can be applied directly to the current frame without performing the temporal compression process. The video decoder reconstructs a video frame from the compressed bit stream. It applies the entropy decoder to coefficients and motion vectors, and uses these motion vectors together with one or more previously decoded frames, to create a prediction of the current frame (Motion Compensation process). By adding the residual frame to this prediction, it obtains the current frame.

The aforesaid model is often described as hybrid DPCM/DCT (Differential Pulse Code Modulation/Discrete Cosine Transform) model also. Figure 2.2 and Figure 2.3 show a generic DPCM/DCT hybrid encoder and decoder respectively [I.E03].

In Figure 2.2, O_n is the current frame, and it is compared with the reference frame F_{n-1}. After the motion estimation process, a set of motion vectors are chosen. Based on these motion vectors, a motion compensated prediction P is generated, and subtracted from the current frame, to produce a residual frame D that is transformed using DCT and quantization (Q). Finally, quantized DCT coefficients and motion vectors are entropy encoded to produce the compressed bitstream. Inverse quantization (IQ) and inverse DCT (IDCT) are applied to quantized DCT coefficients, to produce a decoded residual $D^{'}$ that is not identical to D, because of the quantization process. The residual $D^{'}$ is added to the motion compensated prediction P, to produce the reconstructed frame F_n that may be used as a reference frame for the next encoded frame F_{n+1}. In Figure 2.3, the compressed bitstream is entropy decoded to extract coefficients and motion vectors. Inverse quantization and inverse

DCT are applied to quantized DCT coefficients to produce a decoded residual D'. Decoded motion vectors are used to perform motion compensation in the reference frame F_{n-1}. The result is the motion compensated prediction P that is added to D' to produce the decoded frame F_n that can be displayed and may be stored as a reference frame for the next decoded frame F_{n+1}. We will explain in more detail in the next sections operations performed in this codec model.

2.3.1 Motion Estimation

The goal of the temporal model is to reduce redundancy between transmitted frames, by forming a predicted frame, and subtracting this from the current frame, to obtain a residual frame. The predicted frame is created from one or more past or future frames (reference frames). The simplest method of temporal prediction is to use the previous frame as the predictor of the current frame. The more accurate is the prediction process, the less information is contained in the residual frame. Much of this residual information is due to object movements between two consecutive frames. The most practical and widely used method for temporal prediction is motion estimation, together with motion compensation (that we will explain below) on rectangular sections or blocks of $M \times N$ samples of the current frame. The motion estimation, performed in the video encoder, is the procedure able to find a $M \times N$ samples region, in the reference frame (previously coded and transmitted), matching the $M \times N$ block in the current frame. This is carried out by comparing the $M \times N$ block in the current frame with some or all the possible $M \times N$ regions in the search area (a region centered on the current block position), and finding the region that gives the "best match". A way to determine the "best match" is to consider the information in the residual formed by subtracting the candidate region from the current $M \times N$ block. The candidate region that minimizes the residual information is chosen as the best match. The chosen candidate region is subtracted from the current block, to form a residual $M \times N$ block, which is coded. The offset

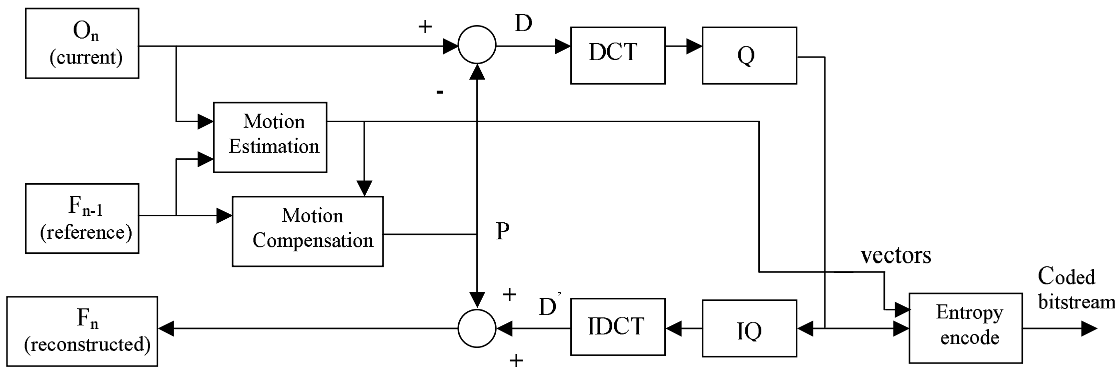


Figure 2.2: DPCM/DCT video encoder

(called motion vector) between the current block and the position of the candidate region is given back and coded. A good choice of the candidate region minimizes the information in the residual frame, improving compression performance.

Accuracy and computational complexity of the motion estimation process depend on the measure adopted for computing the residual information, and the total number of computing times (the last one is related to the range of the search area). The most important measures adopted to compute the residual information are Mean Squared Error (MSE), Mean Absolute Error (MAE), Sum of Absolute Errors (SAE), that we list in the following:

$$MSE = \frac{1}{NxM} \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} (C_{ij} - R_{ij})^2 \tag{2.1}$$

$$MAE = \frac{1}{NxM} \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} |C_{ij} - R_{ij}| \tag{2.2}$$

$$SAE = \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} |C_{ij} - R_{ij}| \tag{2.3}$$

where $M \times N$ is the block size and C_{ij} and R_{ij} are the current and reference area samples respectively. SAE is the most widely-used measure of residual information for its computational simplicity. An exhaustive search in the motion estimation (called Full Search motion estimation) evaluates the information residual measure at each point in the search area. Full Search estimation is guaranteed to find the minimum value of residual information measure in the search area, but it is computationally intensive, since the information measure must be computed at every one of $(2S + 1)^2$ locations, where S is the range of the search area. In a typical video sequence, most motion vectors are equal to zero, so it is likely that the minimum information measure will be found around the (0,0) point. The computation of Full Search motion estimation can be simplified by starting the search at (0,0) point, and proceeding to test points in a spiral pattern around this location, by terminating early the search when the previous minimum information value has been exceeded. In many applications the so-called "fast search" algorithms are used. Many fast search algorithms have been proposed, such as Logarithmic

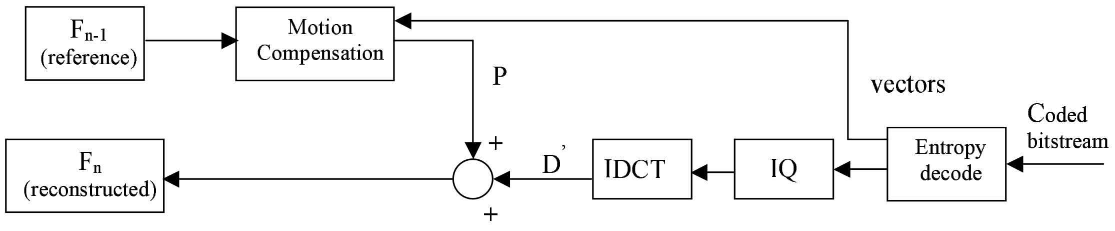


Figure 2.3: DPCM/DCT video decoder

Search, Hierarchical Search, Cross Search, Three Step Search, Nearest Neighbours Search [Gha90][GCK99]. These algorithms compute the information measure for a subset of locations within the search area. Their performance can be evaluated by comparison with the Full Search, in terms of computational complexity and video quality. A better performance of the motion estimation process can be achieved by using fractional values (sub pixels) rather than integer values of motion vectors only. Half-pixel and quarter-pixel values of motion vectors are used in MPEG4 and H.264, respectively. Sub-pixel search requires interpolation between integer samples positions in the reference frame, that is computationally intensive. It is not common to compute sub-pixel samples for the entire search area. To find the best integer-pixel match with Full Search or one of the fast search algorithms is sufficient, and then to search interpolated positions, adjacent to this position.

2.3.2 Motion Compensation

The chosen predicted region by the motion estimation process in the reference frame is subtracted from the $M \times N$ current block to produce a residual frame (luminance and chrominance). This operation, performed in the video encoder, is called motion compensation. The residual frame is coded and transmitted together with the motion vector. The decoder uses the received motion vector to recreate the predicted region, decodes the residual block, and adds this one to the predicted region, reconstructing a version of the original block. In addition, in the encoder, the residual frame is decoded, and added to the predicted region, to form a reconstructed block which is stored as a reference for further motion-compensated prediction. The encoder uses the reconstructed block to perform the prediction of next block so that it can have the same reference for motion compensation that the decoder uses to reconstruct the original block.

In many coding standards, including MPEG2, MPEG4, H.263, and H.264, the basic unit for motion compensated prediction is the macroblock, corresponding to a 16×16 pixel region. Smaller motion compensated block sizes (8×8 or 4×4 pixel regions) can produce better motion compensation results, reducing the residual information. However, such smaller block sizes increase the complexity of the motion compensation process, in terms of search operations and bits to encode an increased number of motion vectors. H.264 codec adapts the motion compensation block size according to the frame characteristics, so that large block sizes are adopted in homogeneous regions of a frame, while small block sizes are chosen around areas with large motion and great details. This method of partitioning a macroblock into motion-compensated sub-blocks of varying size (two 16×8 partitions or two 8×16 partitions or four 8×8 partitions, each one of these 8×8 partitions may be split in a further four ways, or as one 8×8 sub-macroblock, two 8×4 partitions, two 4 x 8 partitions or four 4×4 partitions) is known as *Tree Structured motion compensation* [I.E03]. The motion compensation, as well as the motion estimation process, can be performed with a sub-pixel resolution,[1] by searching sub-sample interpolated positions, as well as integer-sample positions, selecting the position that gives the minimum residual information, and using the integer or sub-sample values at this position for motion compensated prediction.

2.3.3 DCT

The most popular transformation adopted in the spatial compression model is Discrete Cosine Transform (DCT)[ANR74]. The Discrete Cosine Transform operates on $N \times N$ blocks (usually, N=8 or N=4) of the image or residual samples and creates an $N \times N$ block of coefficients. In particular, the DCT of an $N \times N$ sample block is given by:

$$Y = AXY^T \tag{2.4}$$

and the inverse DCT (IDCT) is given by

$$X = A^T Y A \tag{2.5}$$

where X is a matrix of samples, Y is a matrix of coefficients and A is an $N \times N$ transform matrix. The elements of A are:

$$A_{ij} = C_i \cos \frac{(2j+1)i\pi}{2N} \tag{2.6}$$

where:

$$C_i = \begin{cases} \sqrt{\frac{1}{N}} & \text{if i=0} \\ \\ \sqrt{\frac{2}{N}} & \text{if i>0} \end{cases}$$

For example the transform matrix A for a *4* $\times$ *4* DCT is:

$$\mathcal{A} = \begin{pmatrix} \text{a} & \text{a} & \text{a} & \text{a} \\ \text{b} & \text{c} & \text{-c} & \text{-b} \\ \text{a} & \text{-a} & \text{-a} & \text{a} \\ \text{c} & \text{-b} & \text{b} & \text{c} \end{pmatrix}$$

where

$$a = \frac{1}{2}$$

[1]The term sub-pixel is used in this setting even if the motion estimation and compensation process is applied to luminance and chrominance samples, not pixels.

$$b = \sqrt{\frac{1}{2}}\cos(\frac{\pi}{8})$$

$$b = \sqrt{\frac{1}{2}}\cos(\frac{3\pi}{8})$$

The basic concept of this reversible transformation is that, by having the coefficients in the discrete cosine transform domain, it allows marking the visual information in an image, that the human eye is unable to perceive. Such information is deemed redundant and can be discarded without introducing noticeable visual effects. The more significant DCT coefficients are the "low frequency" positions, clustered around the top left coefficient. "Higher frequency" DCT coefficients are very often quantized to zero. After DCT transformation and quantization, the coefficients are reordered to group together the nonzero ones. The optimum scan order, depending on the distribution of nonzero DCT coefficients, can be for example, a zigzag scan starting from the top left coefficient to create one dimensional array with nonzero coefficients within its first positions, followed by long sequences of zero values. To compact the large number of zero values, it is possible to apply a "run-level encoding", that allows to represent this array as a series of (*run*, *level*) pairs where *run* indicates the number of zeros preceding a nonzero coefficient and *level* indicates the number of nonzero coefficients. These data are the input for entropy encoding.

Like other block based transformations, the Discrete Cosine Transform has low memory requirements, but tends to suffer from artefacts at block edges. The Discrete Wavelet Transform (DWT or "wavelet") [CL99] operates on the entire image, and outperforms the block-based transformation, but it has higher memory requirements, since the entire image is processed as a unit, and it can not be used with a block based motion compensation process. JPEG 2000 was developed to provide a more efficient successor to the original JPEG. It uses the "Discrete Wavelet Transform" technology as its basic coding method and provides superior compression performance to JPEG avoiding the characteristic blocking artefacts of a DCT-based compression method.

2.3.4 Quantization

Quantization is a process that maps a signal with a range of values to a quantized signal with a reduced range. This allows to represent the quantized signal with fewer bits than the original one, since the range of possible values is smaller. Most video coding standards assume a scalar quantization as in the following:

$$Z_{ij} = round(Y_{ij}/Qstep)$$

where Y_{ij} is a coefficient, *Qstep* is a quantization step size and Z_{ij} is a quantized coefficient. Round is the rounding operation. The values of *Qstep* are indexed by a Quantization Parameter (QP) and are correlated to the position *(i, j)* of the coefficient in the image. In video compression codecs, the quantization operation (Q) is made in the encoder. A higher range of quantization steps makes it possible for an encoder to have a more accurate control of the tradeoff between bit rate and quality. If the step size is large, the range of quantized values is small, the image is greatly compressed, but the re-quantizated values are very different from the original ones. On the contrary, if the step size is small, the re-quantized values match more closely the original ones, but the larger range of quantized values reduces the compression efficiency.

2.3.5 Entropy Coding

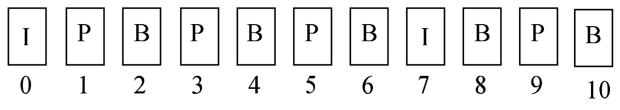
The entropy coding converts a set of symbols (representing quantized transform coefficients after run-level encoding and motion vectors) in a compressed bitstream. Two of the most common entropy encoding techniques are modified “Huffman” variable length coding, and arithmetic coding. In the first one, a variable length coding maps input symbols to a series of codewords (variable length codes or VLCs) so that frequently-occurring symbols are represented with short codes, while less common symbols are represented with long codes. After that, Huffman coding assigns a variable length code to each symbol based on the probability of occurrence of different symbols [Huf52]. Arithmetic coding is a practical alternative to Huffman coding [WNC87]. It is a form of variable length entropy encoding, where the entire message is coded into a single fractional number between zero and one. An efficient arithmetic coding system is Context-based Adaptive Binary Arithmetic Coding (CABAC), used in H.264 codec. It achieves good compression performance by selecting probability models for each syntax element from a selection of available models depending on the statistics of recently-coded data symbols. It uses the arithmetic coding, updating the selected context-based probability model according to the actual coded value. [MWS03].

2.4 Video coding features

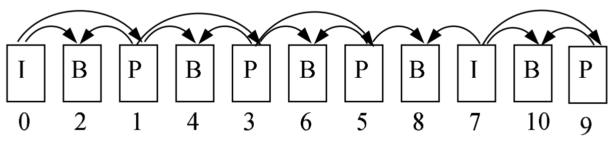
In this section we address some video coding concepts that will be referred in the next chapters.

2.4.1 Types of frames

Different coding standards provide a different format of coded video sequence. In all coding standards, a coded frame consists of a number of macroblocks (basic unit of motion information), each one containing 16 x 16 luminance samples, and associated chrominance samples (8 x 8 Cb and 8 x 8 Cr). There are three main types of macroblocks:



(a) Decoding order



(b) Display order

Figure 2.4: Typical MPEG4 video frame pattern

- INTRA macroblock, which is coded without motion reference to previous or successive frames. In MPEG4, it is coded by DCT, quantization and entropy coding, while in H.264 it can be as well predicted from previously coded data within the same slice (a set of macroblocks). A frame with all macroblocks coded as intra macroblocks is an INTRA frame (*I* frame);
- INTER macroblock, which is predicted by one or more previous reference frames. A frame with macroblocks of type intra and inter is called INTER frame (*P* frame);
- BIDIRECTIONAL macroblock, which is predicted by previous and future frames. A frame containing bidirectional macroblocks is called bidirectional frame (*B* frame). A *B* frame can contain intra and inter macroblocks, also.

When there is a great change between the reference and the current macroblock (scene change), it can be useful to encode the macroblock in intra mode (without motion compensation). A typical coded video sequence includes all three types of frames and a fixed frame type mixing pattern is repeated throughout the entire video sequence. An example of MPEG4 video frame type mixing pattern, named Group of Pictures (GOP) is shown in Figure 2.4, where the arrows indicate the prediction directions. Note that, due to the presence of *B* frames, the decoding order is different from the display order of the video sequence.

2.4.2 Rate Control

If the control parameters (motion estimation search area and quantization step size) of a video encoder are kept constant, the number of bits produced for each frame will change depending on the content of the frame (more bits when there is more motion and more details, fewer bits when there is low motion and not much details). This causes a variable bit-rate (measured in bits per second) of the encoder output. This variable bit rate cannot be supported, for example, by a constant bit-rate channel. Moreover, it is needed to adapt the bit-rate produced by a video encoder to match the available bit rate of the transmission mechanism.

A typical technique used by the encoder to smooth variable bit-rates is buffering the encoded data prior to transmission. This buffer is emptied at a constant rate according to the channel capacity. A similar buffer is placed at the input of the decoder, and it is filled at the channel bit rate and emptied at variable bit rate. High bit rate variations can determine over-or-under flowing of the buffer, and great decoding delay. Rate control is a mechanism able to control the encoder output bit rate, preventing buffer overflow and underflow. Rate control modifies the quantization parameters (QP), since increasing QP reduces bit rate with a lower quality of the decoded frames. The focus of rate control is to optimize the trade-off between the encoder output bit rate and the quality of the decoded video sequence. Many rate control algorithms for video coding have been proposed, according to different video applications and coding standards [CL99][OK95][KH04][DL96] [LCZ97][SALZ06][SA04][MGL05].

2.5 Video quality measure: PSNR

The so-called quality measures are used to evaluate the quality of a video sequence. Video quality measurement is an important requirement for designers and developers of video communication systems. Quality measurement is especially important for compressed video, since most video compression algorithms are lossy, i.e. compression is achieved at the expense of video quality.

Video quality measurement methods can be categorized as *objective* (based on automatic measurement of parameters of the video sequence) or *subjective* (based on evaluation by one or more human observers)[RK04]. Subjective measurements can provide a realistic guide to the video quality perceived by a user. However, obtaining an accurate outcome from subjective tests typically involves many repetitions of a time-consuming test with a large number of test subjects. For this, they are expensive and difficult to carry out. Objective measurements are automatic and readily repeatable. Sophisticated objective quality models have been proposed in [TG00][DKD06]. However, the complex nature of human visual perception makes difficult accurately modeling the response of a human observer. There are not yet objective measurement systems that completely reproduce the subjective experience of a human observer watching a video display.

The most widely used measure in literature, is Peak Signal to Noise Ratio (PSNR). It is an objective measure, computed on a logarithmic scale as described in the following

$$PSNR_{dB} = 10log_{10}\frac{(2^n - 1)^2}{MSE} \tag{2.7}$$

where $(2^n - 1)^2$ is the square of the highest possible signal value in the frame, (where n is the number of bits per image sample). The mean squared error (MSE) is computed between the original and the reconstructed (coded and decoded) frame. We used PSNR to evaluate the proposed approaches in this thesis.

Chapter 3

Video Transcoding

Abstract

In this chapter, we provide an overview of transcoding goals and issues. In addition, we address transcoding solutions for bit rate reduction, resolution reduction and error resilience purposes. Temporal transcoding, which is the focus of this thesis, will be presented in detail in the next chapters.

3.1 Introduction

Today, in everyday life, for working or practical purposes, there are many different devices able in different ways to access Internet or other kinds of network. The term "pervasive" is typically used for indicating the capability of computing devices to implement and access a variety of services and applications, often involving multimedia information. Such devices comprise personal digital assistants (PDAs), hand-held computers (HHC), smart phones, and also automotive computing devices and wearable computers, allowing ubiquitous access to multimedia information in different ways, by using different wired and wireless networks. Such a setting is usually referred as Universal Multimedia Access (UMA). Multimedia comprises images, video, audio, graphics, text, and the so called presentation of media content.

In this thesis, we focus on delivering of video content. The quality of video sequences is influenced by devices capabilities on communication, processing, storage and display. Adapting the media content to different network characteristics (channel bandwidth and terminal complexity, for instance) is one of the most important problems in this setting. To flexibly deliver multimedia data to users with different available resources, access networks and interests, the multimedia content must be adapted dynamically according to the usage environment.

Transcoding enables personalized media delivery and facilitates efficient use of network resources. Transcoding can be performed both on-line (real-time) and off-line. Video transcoding is defined as a process that dynamically adjusts the features or coding-parameters of pre-encoded video, to match the underlying network-bandwidth and/or end-system constraints. The proposed dynamic adjustments include video format portability, bit rate, frame rate and resolution reductions. Furthermore, with the introduction of packet radio services over mobile access networks, error resilience video transcoding has gained a significant attention. Its aim is to increase the resilience of the original bit-stream to transmission errors.

The present chapter is organized in this way: in Section 3.2, we outline main issues and goals of video transcoding process; in Section 3.3, we provide a high level overview of the techniques and architectures used to improve the performance of transcoding, in particular those ones for bit-rate reduction (Section 3.3.1), and spatial resolution reduction (Section 3.3.2). In Section 3.4 we briefly present some concepts related to transcoding for the error resilience purposes.

3.2 Why video transcoding

The goal of Universal Multimedia Access (UMA) is to enable users, having devices with limited capabilities in communication, processing, storage and displaying, to access a multimedia content according to their requirements or preference, as depicted in Figure 3.1. This goal can be achieved in different ways. The first one is by managing different copies of the same multimedia content, one for each device type: in this manner, there is a waste of storage resources, as well as bandwidth over-usage due to duplicated transmissions. Another way is by having a scalable media model that provides a base layer for minimum requirements, and one o more enhancement layers to offer improved quality at increasing device capabilities. In this model, the overall video quality degrades significantly with the increased level of scalability, particularly when the base layer is encoded at a low bit rate. Furthermore, this strategy needs layered encoding and decoding capabilities at the server and receiver sides, respectively. Low-power mobile terminals do not have such functionalities requiring increased device complexity. The last way, is to adapt the media content *on-the-fly*: by "transcoding". Transcoding is usually performed by servers of a communication system, or by gateways interconnecting different networks.

Transcoding is defined, in general, as the conversion of one compressed signal into another one. Video transcoding is the process of converting a coded video sequence into another one with different features, without totally decoding and re-encoding, so reducing the complexity and the running time, and enabling the interoperability of heterogeneous multimedia networks. Video transcoding can provide format conversion, resolution scaling (spatial transcoding), bit rate conversion (quality transcoding), frame rate conversion (temporal transcoding). Format conversion operates a syntax change from a video coding standard into another one. Spatial transcoding reduces the spatial resolution of a compressed video. It is required for facing the problem of limited size in many access terminals. Quality transcoding operates on the bit allocation for each frame, by tuning the quantization parameters, according

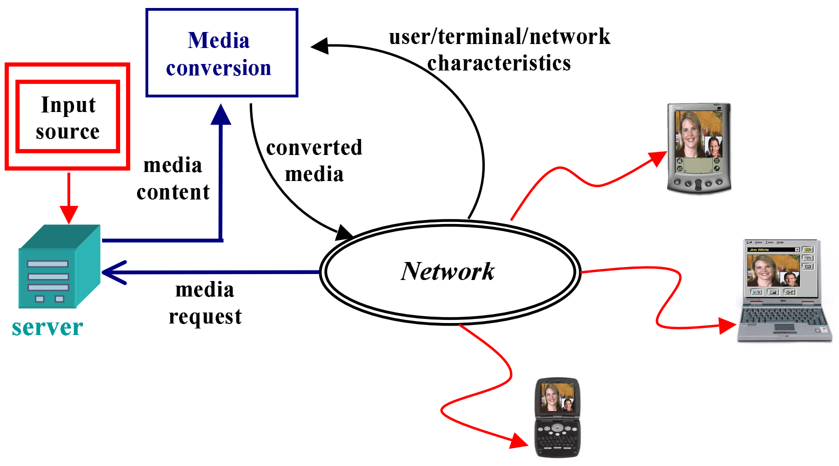


Figure 3.1: Universal Multimedia Access

to the target bit rate. Temporal transcoding is a process that skips some frames in order to change the frame rate of the video sequence, without decreasing the video quality for not skipped frames.

Transcoding strategy helps the content provider to keep only one high quality copy of video stream, reducing the storage costs significantly. Likewise, the operational complexity associated with layered encoding and decoding is not required. This makes transcoding a viable option for content adaptation in heterogeneous networking, particularly for a wide range of mobile terminals with limited processing and battery power. One transcoding application scenario is delivering a high-quality multimedia source to several receivers (such as PDAs, Pocket PCs and desktop PCs) on wireless and wireline networks, where a transcoder can generate an appropriate bitstream directly from the original bitstream, without having to decode and re-encode. To suit available network bandwidth, a video transcoder can perform dynamic adjustment in the bit-rate of the video bitstream, without additional functional requirements in the decoder. Another scenario is a video conferencing system on Internet, in which the participants may use different terminals. In such case, a video transcoder can offer dual functionality: provide video format conversion to enable content exchange, and perform dynamic bit-rate adjustment to facilitate scheduling according to network resources [AWSZ05].

3.3 Video transcoding architectures

The simplest configuration of a system including a transcoder is described in Figure 3.2, where encoder and decoder modules in the transcoder can be replaced by those ones presented in Figure 2.2 and Figure 2.3, respectively. For the sake of clearness, in this thesis, we call "Front Encoder" and "Front Decoder" the encoder and decoder put in input and output to transcoder respectively.

The straightforward approach for implementing transcoding is to cascade a standard decoder and a standard encoder. In this approach, commonly known as cascaded pixel-domain transcoding, the incoming video bitstream is decoded in the pixel domain, and then the decoded video is re-encoded into the target video stream, with the desired format, output bit rate, spatial resolution and frame rate. This approach provides for the best possible quality, but requires a lot of processing and memory resources. Many architectures have been proposed to make more efficient the transcoding process having a minimum impact on the quality of the transcoded video. In the following, we outline such architectures for bit-rate reduction and spatial resolution reduction.

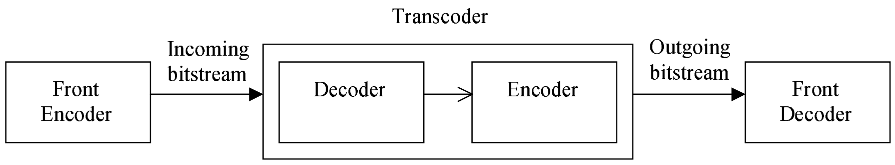


Figure 3.2: Cascaded fully decoding/re-encoding scheme

3.3.1 Bit-rate reduction

The goal of bit rate reduction is to reduce the bit rate while maintaining low complexity and achieving the highest possible quality. A simple technique to transcode a video to lower bit rate is to increase the quantization step at the encoder in the transcoder [NHK95]. This decreases the number of nonzero quantized coefficients, and then the amount of bits in the outgoing bitstream. Such transcoding approach produces a variable frame quality and it is usually know as "quality transcoding".

Many architectures for bit rate reduction suffer from quality degradation caused by the so called "*drift error*". The drift error occurs when the reference pictures reconstructed and stored in the decoder are not the same that those ones used to perform prediction in the encoder. Since the current reconstructed picture is also used to perform prediction for the next frame, the drift error propagates to future frames, and the degradation of video quality increases until an INTRA frame is reached.

The most straightforward way to have a drift-free transcoding is to use a cascaded pixel-domain transcoder (CPDT) architecture, where an encoded bitstream is decoded, and then fully re-encoded at the new bit rate. As explained in [YSX99], since CPDT converts the bit rate of the incoming bitstream by manipulating the content of the bitstream, the reconstructed pictures at the front-encoder and at the front-decoder will not be the same. However, this mismatch does not produce the drift because the decoder-loop and the encoder-loop in the CPDT architecture are completely separated, as illustrated in Figure 3.3. In this architecture, the recon-

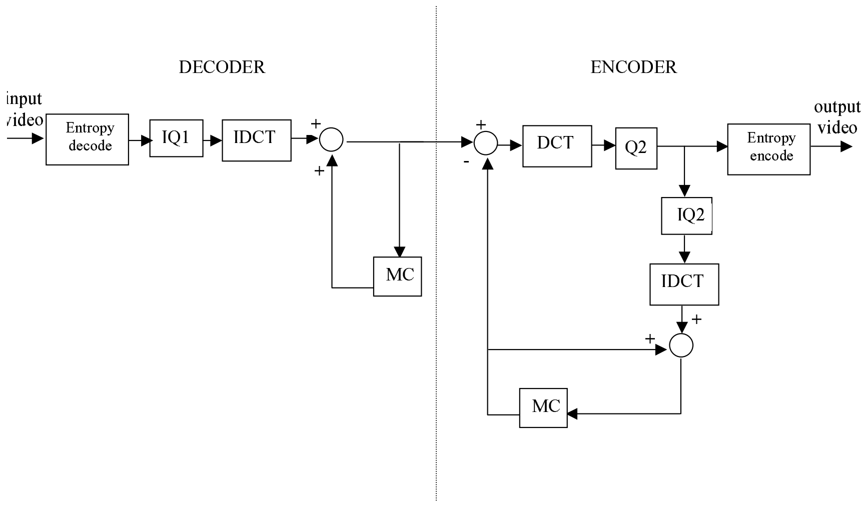


Figure 3.3: Cascaded Pixel-Domain Transcoder (CPDT)

structed pictures in the transcoder's decoder will track the reconstructed pictures in the front encoder, and the reconstructed pictures in the transcoder's encoder will track the reconstructed pictures in the front decoder.

The authors of [YSX99], define the following drift-free condition:

$$P_n^0 = P_n^1 \quad and \quad P_n^2 = P_n^3 \tag{3.1}$$

where P_n^0, P_n^1, P_n^2 and P_n^3 are the pixel values of the n^{th} reconstructed pictures in the front encoder, transcoder's decoder, transcoder's encoder and in the front decoder, respectively.

CPDT architectures satisfy the above drift-free condition. However, the direct implementation of the CPDT is not desirable, since it has a high complexity.

The most simplified architecture for bit rate reduction is the open-loop architecture [VCS03]. In the open-loop system, depicted in Figure 3.4, the bit stream is entropy decoded, to extract the code words corresponding to the quantized DCT coefficients, those ones are inverse quantized (IQ_1) and simply re-quantized with a new quantization parameter (Q_2) to satisfy the new output bit rate. Finally, the re-quantized coefficients are entropy coded. An alternative open-loop scheme directly discards high frequency DCT coefficients of a block.

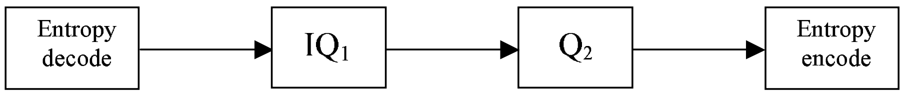


Figure 3.4: Open-loop transcoding architecture for bit rate reduction

In comparison to cascaded pixel transcoders open-loop systems are relatively simpler, since a frame memory is not required, and the operations are performed directly on the DCT coefficients. However, open-loop architectures are subject to drift problem [XLS05]. In general, the reason for drift is due to the loss of high-frequency information. Beginning with the INTRA frame, which is a reference for the next INTER frame, high frequency information is discarded by the transcoder, in order to meet the new target bit-rate. This happens for the residual of an INTER frame also. The front decoder receives such transcoded bitstream, decodes it and stores each frame with reduced quality. This frame is used as predictive component, and it is added to a degraded residual content, leading to drift errors. To avoid the above drift-error, the reference pictures reconstructed and stored in the front decoder must be the same than those ones used to perform prediction in the transcoder's encoder. The open-loop transcoder changes the residual and, therefore, makes the reference picture used in the front decoder different from that one used in the encoder [VCS03]. Closed-loop transcoders contain a feedback loop in the transcoding architecture in order to compensate the drift in transcoder. In [VCS03] the closed-loop architecture depicted in Figure 3.5 is presented.

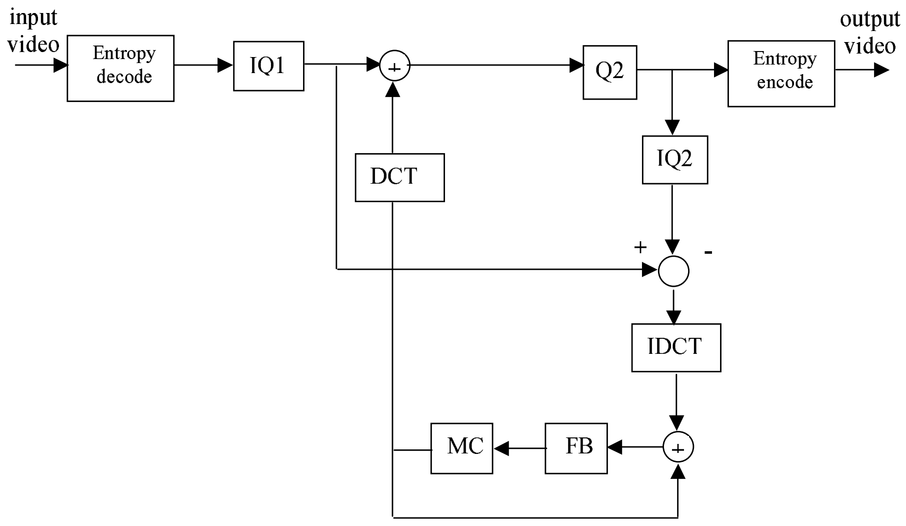


Figure 3.5: Closed-loop transcoding architecture for bit rate reduction

With respect to CPDT architecture (see Figure 3.3), in this simplified scheme only one reconstruction loop is required, with one DCT and one IDCT. In such scheme, some arithmetic inaccuracy is introduced, due to the non linear nature in which the reconstruction loops are combined.

Other schemes reducing the computational complexity of the drift-free CPDT architecture have been proposed [BC98][AG98]. In a fast pixel domain transcoder (FPDT) [BC98], the encoder, rather than performing the full-scale motion estimation, as in the front encoder, reuses the motion vectors extracted from the input video bitstream. Thus, the motion estimation, which usually takes 60-70% of the encoder computation [SG00], is omitted. Since the motion vectors extracted from the input video bitstream were computed at the front encoder for the original quantization parameter (Q_1), they might not be suitable for the new quantization parameter (Q_2). A refinement process in a search window of few pixels (± 3) around the motion vectors extracted from the input video bitstream is needed. Fast search methods can be applied to further reduce the computational complexity of this operation.

Together with motion vectors, macroblock type decision information extracted from the input video bitstream can be reused. This can cause a wrong coding of macroblocks in the transcoder. For instance, if a macroblock has all coefficients equal to zero for a larger quantization parameter, should be coded as SKIPPED macroblock in the transcoder's encoder. If this macroblock was coded as INTER macroblock in the front encoder, sample reusing of block type mode decision information imposes to code this macroblock as INTER macroblock in the transcoder's encoder also. A solution to such problem, proposed in [BC98], is to re-estimate the macroblock type in the transcoder's encoder, as follows:

- If a macroblock was coded as INTRA at the front encoder, it must be coded as INTRA macroblock again;
- If a macroblock was coded as SKIPPED macroblock, it must be coded as SKIPPED macroblock again;
- If a macroblock was coded as INTER macroblock, and all coefficients are equal to zero, it must be coded as SKIPPED macroblock, else it can be coded as INTER or INTRA macroblock.

Note that, SKIPPED macroblocks have no information about motion and residual. They are approximated with the corresponding macroblock of the previous frame at the decoder.

Another source of computational complexity in transcoding is DCT. A simplified DCT-domain transcoder (SDDT) shown in Figure 3.6 was presented in [AG98]. SDDT eliminates the DCT/IDCT, and reduces the memory need. In this architecture, motion compensation (MC) is performed in the frequency domain, and it is the most computation intensive operation. As shown in Figure 3.7, the goal is to derive the DCT coefficients of the target DCT block B, from the coefficients of its four overlapping DCT blocks B_1, B_2, B_3, B_4. The solution proposed in [LL01] to derive the DCT coefficients uses multiplications and additions of constant geometric transform matrices. The fact that the information of a DCT block is concentrated in its low frequency coefficients is exploited to perform an efficient approximation of the MC operation.

The fast architectures (FPDT and SDDT) may require less computation than CPDT but suffer from the quality degradation, caused by the drift problem. This is due to rounding operations performed in the interpolation of fractional-pixel in MC,

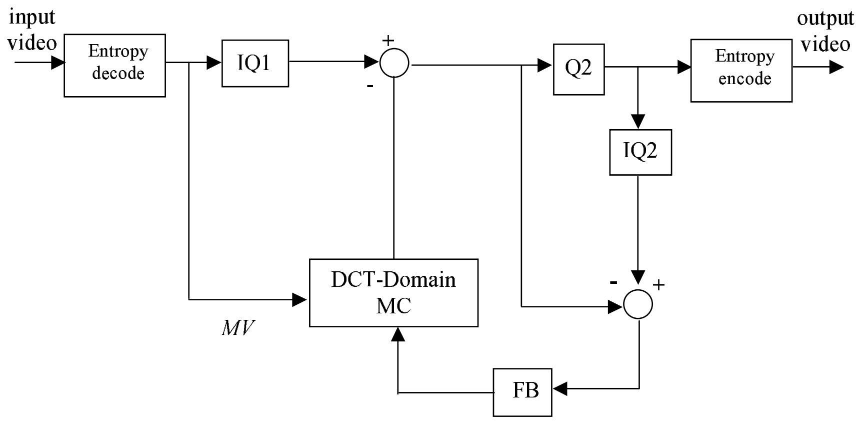


Figure 3.6: Simplified DCT-domain transcoder (SDDT)

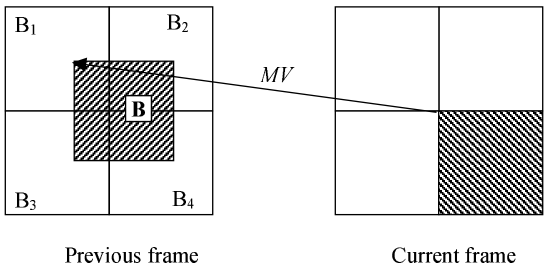


Figure 3.7: DCT-domain Motion Compensation

and to the use of clipping functions to limit the pixel values and DCT coefficients. Detailed analysis of drift causes in fast architectures can be found in [YSX99]. To reduce the drift problem, the front encoder can send frequently INTRA frames. The transmission of INTRA frames requires much more bandwidth than that one of INTER frames. So, it can be not possible when the bandwidth of the outgoing channel is limited.

3.3.2 Spatial Resolution Reduction

The heterogeneity of network access terminals often demands the reduction of compressed video spatial resolution. Besides, reduction in spatial resolution can lower the bit rate. The spatial transcoding reduces the spatial resolution (for example from CIF to QCIF). In a cascaded pixel domain transcoding architecture, similar to that for bit rate reduction, after decoding, the spatial domain down-sampling is applied, followed by a full re-encoding. The most common techniques to reduce spatial resolution are pixel sub-sampling and pixel averaging [SG00].

A simple pixel sub-sampling method down-samples the image by dropping alternate pixel in both horizontal and vertical directions of luminance and chrominance.

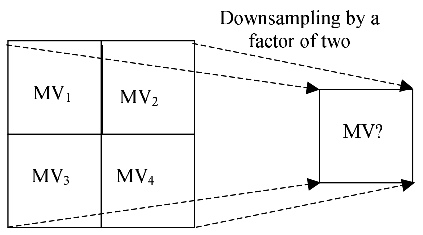


Figure 3.8: Spatial Resolution Reduction: Down-sampling of four motion vectors

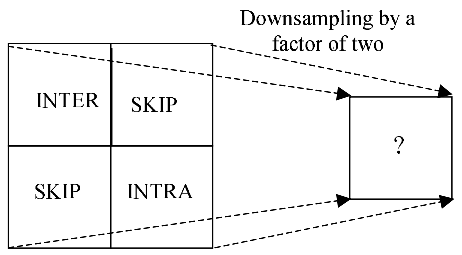


Figure 3.9: Spatial Resolution Reduction: Down-sampling of four macroblock types

Pixel averaging is a technique that represents $m \times m$ pixels by a single pixel which has a value equal to the average of the original $m \times m$ pixel values. Both in pixel sub-sampling and pixel averaging, the critical issue is to map motion vectors of decoded sequence in the new motion vectors for down-sampled sequence. Figure 3.8 shows the problem of merging four motion vectors in one single motion vector, when resolution is reduced by a factor of two in each dimension. The solutions proposed in [BC98] to handle the situation presented in Figure 3.8 are:

- Computing the average of four motion vectors and scaling it;
- Computing the median of three motion vectors and scaling it; The median gives a more accurate result when the motion vector values are very different among them.
- Selecting randomly one of the four motion vectors and scaling it.

In this case scaling motion vector means dividing it by two.

Since these solutions are sub-optimal, a motion vector refinement will be needed at the encoder. In [SZZL01], these strategies are extended to consider transcoding with arbitrary down-sampling ratio by taking into account the unequal contributions of related input motion vectors.

Another problem to be addressed in spatial transcoding is mapping the incoming macroblocks types to the type of transcoded macroblock, as can be seen in Figure 3.9. The authors of [BC98] adopt the following method:

- If there exists at least one INTRA macroblock among the four original macroblocks, the new macroblock is also intra-coded;
- If there exists at least one INTER macroblock and no INTRA macroblocks, the new macroblock is coded as an inter macroblock;
- If all macroblocks have the SKIPPED type, the new macroblock is coded as SKIPPRD macroblock.

In [YVLS02], an analysis of the drift problem and various types of macroblock-level conversions for reduced-resolution transcoding are considered. Based on the results of the presented analysis, four transcoding architectures that attempt to overcome the types of drift errors that have been identified are proposed.

In [AWSZ05] a deeper survey of various methods used to perform spatial resolution reduction is given.

3.4 Video Transcoding for Error-Resilience

The encoding process is typically performed without a prior knowledge about the channel characteristics of network hops between the encoder and the decoder. One problem that any communication system shows is that information may be altered or lost during transmission due to channel noise or transmission collisions. Any damage to the compressed bit stream may lead to visual distortion at the decoder. The authors of [WZ98] review the techniques of error control and concealment developed for video communication over unreliable channels, such as wireless ones. These techniques are classified in three categories, according to the roles that encoder and decoder play in the underlying approaches: *forward error concealment*, which includes methods that add redundancy at the source to enhance error resilience of the coded bit streams; *error concealment by postprocessing* refers to operations at the decoder, to recover the damaged areas based on characteristics of image and video signals. The last, that is *interactive error concealment*, covers techniques that are dependent on a dialogue between source and destination.

In heterogeneous multimedia systems, video transmission may involve wired and wireless channels. Wired channels typically have relatively high bandwidth and consistently low bit error rates. This is in contrast to a typical wireless channel, in which the bandwidth is generally lower and the error rate much greater and time varying. The resilience is the ability of a bitstream to accommodate to the channel conditions and produce acceptable quality. However, this may not be possible for different reasons. First, an encoder designed for use on a wired channel is usually not “wireless aware”, in the sense that it operates without knowledge of any wireless channels between encoder and decoder. Second, in multicast transmissions, an encoder can only produce a single bitstream that is transmitted to different users over different channels.

The transcoding process for error resilience purposes can be performed in a network node (e.g. mobile switch/base station, video gateway). The goal is to modify the already encoded video bit-stream to improve its resilience prior to transmission over a high error rate channel.

Error resilience transcoding has been studied in [RRCC00] and [DCU$^+$02]. In [RRCC00], the authors derive analytical models characterizing how corruption propagates in a video compressed using motion-compensated encoding and subject to bit errors. These models are used to generate the resilience rate-distortion functions used to compute the optimal temporal and spatial resilience to inject into the bitstream. In [DCU$^+$02], an error resilience video transcoding system for transmission of coded video streams over *general packet radio service* (GPRS) mobile access networks is proposed. It uses two resilience schemes, namely *adaptive intra refresh* (AIR) and *feedback control signal* (FCS). The AIR method prevents the error propagation within a video stream, by using a prederminated number of intra macroblocks. The FSC method uses a feedback signal from the receiver, to adapt the encoding schemes. Since both AIR and FSC methods increase the overall transmission rate, the transcoder, tuning the quantization parameters, transforms the output bit rate to adapt it to network conditions.

The error resilience is out of the scope in this thesis. We make the assumption that in our transcoding system the channel is error-free since the error control is faced at lower network levels.

Chapter 4

Temporal Transcoding

Abstract

In this chapter, we introduce the temporal transcoding issues. In particular, we present two temporal transcoding architectures and we outline two problems that, at our best knowledge, are the most important ones in temporal transcoding: motion vector computation and the choice of frames to be skipped. We briefly survey the most common proposed solutions to these problems.

4.1 Introduction

In order to distribute the same encoded video sequence to users through channels with different bandwidths (as for instance in a multicast session), the coded video sequence must be converted into specific bit rates for each outgoing channel. Quality transcoding does this bit-rate conversion by operating on the bit allocation for each frame and by tuning the quantization parameters of every macroblock of the frame according to the target bit rate. The consequence of this is a variable frame quality.

When the bandwidth in a mobile network is very limited, the quality transcoding process can cause high degradation of the transcoded video quality, if the frame rate is constant. Temporal transcoding can be a good alternative to quality transcoding since, by eliminating some frames in the sequence, it reduces the frame rate and the bit rate of the video sequence, without decreasing the video quality of not skipped frames. Different temporal transcoding architectures have been proposed [FCS04] [LH03] [PK05b] [ZZYW05] [ZZYW05]. We will briefly survey some of them in Section 4.2. To reduce the processing complexity and delay, the proposed approaches try reusing the information of the incoming video bitstream, in particular motion vectors and residual information. The main problem in temporal transcoding is that, when some incoming frames are dropped, the incoming motion vectors (MVs) and residual information are not valid, since they refer to skipped frames which do not exist in the transcoded bitstream. Motion vectors and then residual information for the outgoing video stream can be obtained by applying a Full Search Motion Estimation (Section 2.3.1).

In transcoding, Full Search motion estimation is usually not performed, because of its computational complexity. Instead, motion vectors extracted from the incoming bitstream are reused. Reusing incoming motion vectors by composing them to find the motion vector for the outgoing video stream is known as Motion Vector Composition (MVC). We will describe the most important motion vector composition algorithms for temporal transcoding in Section 4.3. However, a simple reuse scheme of incoming motion vectors may introduce considerable quality degradation, as pointed out in [BC98][YSL99][YSL98]. The reconstruction error causes incoming motion vectors to deviate from optimal values. Furthermore, the composition of incoming motion vectors increases such deviation. In most macroblocks, the deviation is within a small range, and the position of the optimal motion vector can be easily obtained by applying a refined motion estimation procedure with a search range of few pixels (±2 or ±3 pixels, in most cases ±1 gives satisfying results) around the base motion vector. Full Motion Estimation or fast search methods can be applied to perform this procedure. In [YSL99] and [CCP02], two motion vectors refinement schemes are proposed, and will be described in Section 4.4.

An important issue in temporal transcoding is how to choose the frames to be skipped, in order to have a good quality of the transcoded video sequence. At our best knowledge, the most important factor addressed in literature in the choice of frames to be skipped has been the motion information contained in a frame. According to the target applications, other frame skipping metrics have been developed. They are based on transcoder buffer constraints or frame content. We give a description of the most important frame skipping policies in Section 4.5.

4.2 Temporal Transcoding Architectures

A temporal transcoder receives as input a coded video sequence, and by dropping some frames, produces as output a video sequence with a reduced frame rate and/or bit rate. After skipping a frame in temporal transcoder, it is needed to reconstruct the next frame according to the last not skipped frame. The most popular approaches to perform this are known as pixel-domain and DCT domain reconstruction.

A pixel-domain temporal transcoder architecture is described in Section 4.2.1. It is a drift-free cascaded pixel-domain transcoder architecture, that is the most favorable in terms of picture quality.

The authors of [FCS04] explain how the re-encoding operation in the pixel domain transcoder architecture introduces error when a frame is skipped. They show that this re-encoding error affects the video quality of non skipped frames, with a high PSNR degradation, compared to that of the same pictures directly decoded

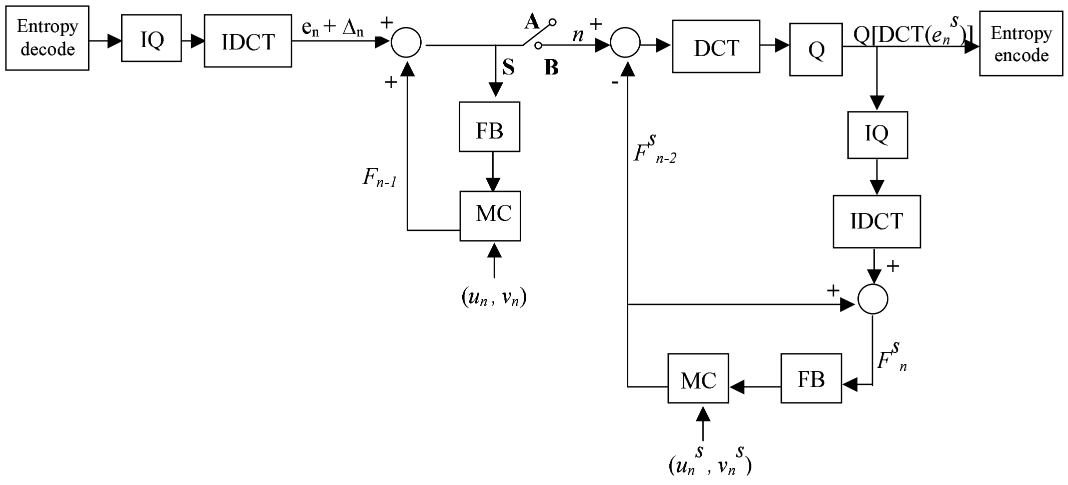


Figure 4.1: Cascaded Temporal Transcoder in the pixel-domain

without the transcoding process. The same authors in the same work, propose a new frame skipping architecture that performs operations on the discrete cosine transform (DCT) domain. Such transcoding architecture, described in Section 4.2.2, achieves low complexity, and avoids re-encoding errors when the strategy of direct sum of DCT coefficients is employed. We present both architectures in the following.

4.2.1 Pixel-domain temporal transcoder

Figure 4.1 shows the structure of a drift free conventional cascaded pixel temporal transcoder [FCS04]. It reduces the output bit rate by skipping frames. The switch S is used to control the desired frame rate of the transcoder, its position coinciding with A if the frame is skipped, and with B if the frame is not skipped, respectively. Note that, skipped frames must also be decompressed completely and they act as reference frame for the reconstruction of not skipped frames in the following way:

Assume that frame F_{n-1} is skipped. However, frame F_{n-1} is required to act as reference frame for the pixel reconstruction of frame F_n, such that

$$F_n(i,j) = F_{n-1}(i+u_n, j+v_n) + e_n(i,j) + \delta_n(i,j) \tag{4.1}$$

where (u_n, v_n) are the horizontal and vertical components of motion vector MV_n, computed at the front encoder for a macroblock with $N \times N$ pixels in the original uncompressed frame O_n, by applying motion estimation in the previously reconstructed frame F_{n-1}. $O_n(i,j)$ and $F_{n-1}(i,j)$ represent a pixel in O_n and F_{n-1} respectively, $\delta_n(i,j)$ represents the reconstruction error of the current frame in the front encoder due to the quantization, $e_n(i,j)$ is the residual between the current frame and the motion-compensated frame. We have that

$$e_n(i,j) = O_n(i,j) - F_{n-1}(i+u_n, j+v_n). \tag{4.2}$$

Substituting (4.2) in (4.1) we obtain that

$$F_n(i,j) = O_n(i,j) + \delta_n(i,j) \tag{4.3}$$

where O_n and F_n are the original and reconstructed frame in the pixel domain. In the transcoder, after skipping frame F_{n-1}, it is needed to re-encode the frame F_n with respect to F_{n-2}, which is the last not skipped frame, since F_{n-1} does not exist anymore. The frame F_{n-2} acts as reference, (instead of F_{n-1}), for frame F_n. By applying a new motion estimation or, as stated above, a MVC algorithm (Section 4.3), it is possible to find the new motion vector (u_n^s, v_n^s) pointing to frame F_{n-2}. The superscript "s" is used here to denote the symbol after performing the frame skipping transcoding. The reconstructed pixel in the current frame F_n after the front-decoder is

$$F_n^s(i,j) = F_{n-2}^s(i+u_n^s, j+v_n^s) + e_n^s(i,j) + \delta_n^s(i,j) \tag{4.4}$$

where $e_n^s(i,j)$, representing the re-quantization error due to re-encoding in the transcoder, is

$$e_n^s(i,j) = F_n(i,j) - F_{n-2}^s(i+u_n^s, j+v_n^s). \tag{4.5}$$

Substituting (4.5) in (4.4), we have that

$$F_n^s(i,j) = F_n(i,j) + \delta_n^s(i,j). \tag{4.6}$$

This equation implies that the reconstructed quality of the not skipped frame deviates from F_n, which is the input sequence to the transcoder. Re-encoding of the current frame involves recomputation of the residual frame between the current frame and the not skipped reference frame, followed by DCT transformation and quantization process. This re-encoding procedure leads to error δ_n^s. The authors of [FCS02][FCS04] propose a DCT domain transcoding architecture where this error is avoided, when the direct sum of DCT coefficients for no motion compensated macroblocks is applied. We describe such approach in the next section. Note that, no motion compensated macroblocks are macroblocks with motion vector equal to zero. Motion compensated macroblocks have motion vector with not zero value. For the sake of brevity, in the following in this thesis, we indicate no motion compensated macroblocks as no-MC macroblocks and motion compensated macroblocks as MC macroblocks.

In [VYLS02], a spatio-temporal transcoding architecture is proposed, where an intra-refresh technique is used to correct the error introduced in transcoding by incorrect motion vector mapping, residual mapping and down-sampling. This technique provides an inter-intra conversion of a block according to properly tuned thresholds. A similar technique is used in the temporal transcoding architecture that we will present in Chapter 5.

4.2.2 DCT-domain temporal transcoder

The DCT-domain transcoding is performed in the coded domain, where complete decoding and re-encoding are not required, with a significantly reduced processing complexity. In temporal transcoding, when one or more reference frames are skipped from the incoming bit stream, re-computing quantized DCT coefficients for the residual of not-dropped frames with respect to the past reference frames is needed. This operation can be performed in the DCT domain. A partial DCT-domain frame skipping transcoder has been proposed in [LH03], where quantized DCT coefficients of residual for no-MC and MC macroblocks are selectively computed in DCT and pixel domain respectively.

A fully DCT-domain frame skipping transcoder has been proposed in [PK05b], where the quantized DCT coefficients for residual of MC macroblocks are also computed in DCT domain, using block translation technique.

A more complex approach is presented in [FCS04]. It is based on the direct sum of DCT coefficients for no-MC macroblocks. In this approach, when a macroblock is no motion compensated, the DCT coefficients of the residual are given by

$$Q[DCT(e_n^s)] = Q[DCT(e_n)] + Q[DCT(e_{n-1})] \tag{4.7}$$

where $Q[DCT(e_n^s)]$ are the quantized coefficients of the residual for the current macroblock I^n with respect to the last not skipped frame F_{n-2}. They are computed directly in the DCT domain by summing $Q[DCT(e_n)]$ and $Q[DCT(e_{n-1})]$ that are the quantized DCT coefficients of the residual for the macroblock I^n in the current frame F_n, and the macroblock I^{n-1} in the reference skipped frame F_{n-1}, respectively, as we show in Figure 4.2 where I^n and I^{n-1} are I_2^n and I_2^{n-1} respectively. Since the macroblock I^n is no motion compensated, its motion vector is equal to zero , so it points to a macroblock in the previous skipped frame, this implies that the quantized DCT coefficients $Q[DCT(e_n)]$ and $Q[DCT(e_{n-1})]$ 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 re-quantization is not necessary for no motion compensated macroblocks, re-encoding errors δ_n^s (mentioned in Section 4.2.1) are also avoided. Many real-world image sequences have a smooth motion that varies slowly, so most of the macroblocks are no-MC [FCS04]. By using a direct addition of the DCT coefficients in temporal transcoder, the computational complexity and the re-encoding errors are significantly reduced for sequences containing a lot of no-MC macroblocks.

For motion compensated macroblocks (MC macroblocks), direct addition of the quantized DCT coefficients can not be used. As we can see in Figure 4.3, the

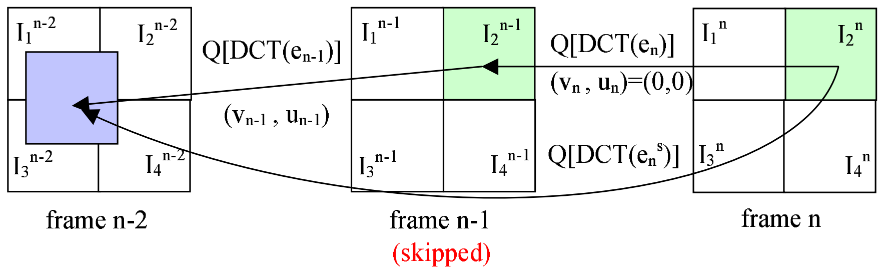


Figure 4.2: DCT-domain temporal transcoder: residual signal re-computation for no-MC macroblocks

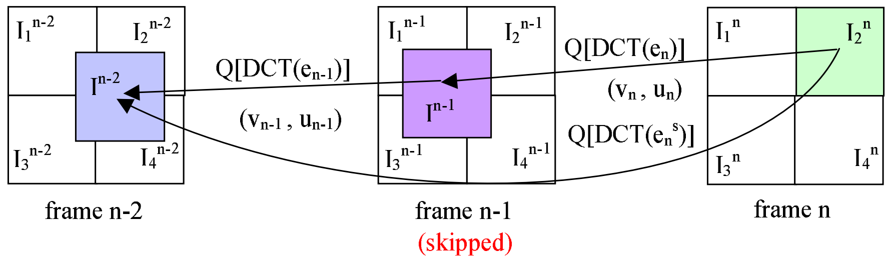


Figure 4.3: DCT-domain temporal transcoder: residual signal re-computation for MC macroblocks

reference area I_{n-1} in the skipped frame is not on a macroblock boundary, and so $Q[DCT(e_{n-1})]$ is not available from the incoming bitstream.

It is possible to use the incoming quantized DCT coefficients of the macroblocks I_1^{n-1}, I_2^{n-1}, I_3^{n-1}, I_4^{n-1} that overlap I^{n-1}, for computing the residual for I^{n-1}, named e_{n-1}. First, inverse quantization and inverse DCT of coefficients of the macroblocks that overlap I^{n-1} are performed, to obtain their corresponding residual in the pixel-domain. These residual are added to the motion compensated segments of the previous not skipped frame to obtain I^{n-1} in pixel-domain. The residual e_{n-1} is obtained by subtracting I^{n-1} from the corresponding motion compensated reference area I^{n-2} in the previous not skipped frame. DCT and quantization are applied to e_{n-1} to obtain $Q[DCT(e_{n-1})]$. Then, the new quantized DCT coefficients $Q[DCT(e_n^s)]$ of a motion compensated macroblock can be computed according to (4.7). The re-quantization introduced for computing $Q[DCT(e_{n-1})]$ brings to additional re-encoding errors δ_{n-1}^s. Note that, as compared with δ_n^s in (4.6), δ_{n-1}^s is the re-encoding due to frame $n-1$ instead of frame n. These errors degrade the quality of the reconstructed frame. Since each not skipped inter-frame is used as a reference frame for the next not skipped inter-frame, quality degradation propagates to later frames in a cumulative manner. In the solution proposed in [FCS04] [FCS02], these re-encoding errors are stored and added to the residual of motion compensated macroblocks in the successive INTER frames. This technique cannot entirely avoid the propagation of re-encoding errors, but it reduces their effect on the visual quality of

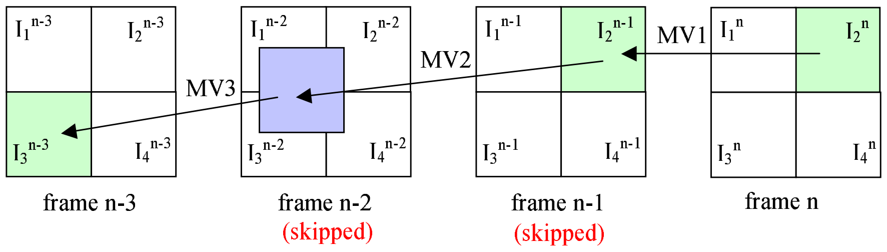


Figure 4.4: Motion Vector Composition

the transcoded frames.

In [ZZYW05], a skipping scheme for frame rate reduction is proposed. It performs operations in the DCT domain and adopts a macroblock coding mode re-judgment algorithm in order to reduce the re-encoding error due to skipping of frames. This algorithm takes into account the macroblock coding mode in the input bitstream and the coding mode of macroblocks overlapping the reference area in the skipped frame.

A macroblock coding mode re-judgment algorithm is adopted also in the transcoder architecture proposed in this thesis, as we will show in Section 5.3.

In [KPKK06], a hybrid quality/temporal transcoder in DCT domain is proposed. It uses a *complexity measure* of a frame to determine whether a frame has to be skipped or not. This complexity measure takes into account the number of generated bits and the average quantization parameter of a frame. It is compared with a threshold, dynamically tuned according to activities of frames in a window and to the target frame rate. In addition, a frame layer rate control method is adopted to perform optimal bit allocation at frame level in order to minimize the average distortion over an entire sequence. In this bit allocation, buffer fullness is considered to meet the target bit rate.

In this thesis, we focus on purely temporal transcoding, where bit rate reduction is obtained only by skipping frames.

4.3 Motion Vector Composition

When some incoming frames are dropped in transcoding, new motion vectors for the outgoing bitstream need to be computed. One possible way to generate the motion vectors of the outgoing sequence, without performing motion estimation, is to use the vector sum. Figure 4.4 illustrates a situation where two frames are dropped. The estimated MV for the macroblock I_2^n in the current frame F_n is the sum of motion vectors MV_1, MV_2, MV_3.

If the incoming motion vector is equal to zero, it points to a macroblock (I_2^{n-1}), so the motion vector of this macroblock (MV_2) is available in the incoming bitstream. In general, motion vectors are not equal to zero, and they point not to a macroblock, but to a reference area of 16×16 pixels which is not on a macroblock boundary. So, the motion vector of this area (MV_3 in Figure 4.4) is not available in the incoming bitstream.

There are several algorithms (Bilinear Interpolation, Forward Dominant Vector Selection, Telescopic Vector Composition, Activity Dominant Vector Selection are the most popular) able to select a motion vector for this reference area, when it overlaps four macroblocks in the skipped frame. Such algorithms, that we describe in the following, are of interest in this thesis. In Section 5.4, we will compare their performance relative to our transcoder architecture implementation.

In H.264 codec, these algorithms cannot be directly applied, due to the variable partitioning of a macroblock into motion-compensated sub-blocks of varying sizes (see Section 2.3.2), since a separate motion vector is required for each partition or sub-block. In this thesis, we approach the above problem, and give an H.264 motion vector composition scheme, described in Section 5.5.2.

4.3.1 Bilinear Interpolation

In [HWL98], bilinear interpolation is defined as:

$$MV_{int} = (1-\alpha)(1-\beta)MV_1^{n-1} + (\alpha)(1-\beta)MV_2^{n-1} + (1-\alpha)(\beta)MV_3^{n-1} + (\alpha)(\beta)MV_4^{n-1} \tag{4.8}$$

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

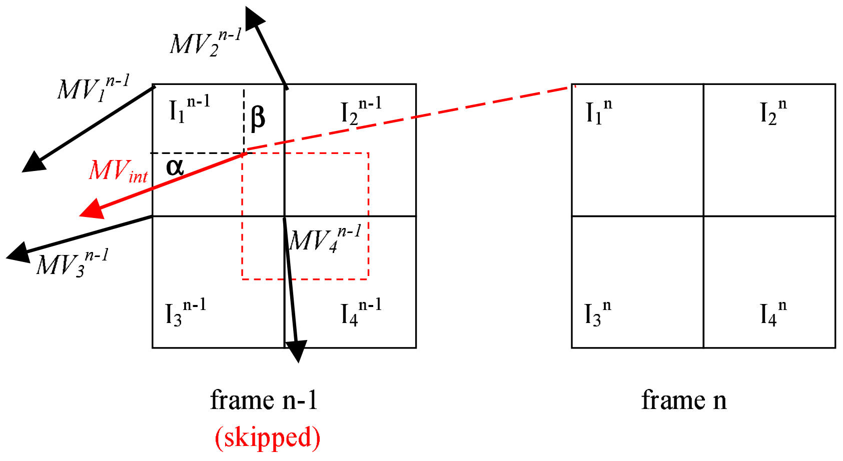


Figure 4.5: Bilinear Interpolation algorithm

macroblocks overlapping the reference area in the skipped frame pointed by the incoming motion vector, α is the ratio between horizontal pixel distance of this reference area from the MV_1^{n-1} and the macroblock size, and β is the ratio between vertical pixel distance of this reference area from the MV_1^{n-1} and the macroblock size (Figure 4.5). The selected motion vector is MV_{int}.

4.3.2 Forward Dominant Vector Selection

In [YSL99] [YS99], the Forward Dominant Vector Selection algorithm is proposed. This algorithm selects one dominant motion vector among the vectors of the four macroblocks overlapping 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 having the largest overlapping area with the reference area pointed by the incoming motion vector. In particular, this dominant vector, MV_{fdvs}, is selected as follows:

$$MV_{fdvs} = MV_1^{n-1} \text{ if } \alpha < 0.5 \text{ and } \beta < 0.5$$

$$MV_{fdvs} = MV_2^{n-1} \text{ if } \alpha \geq 0.5 \text{ and } \beta < 0.5$$

$$MV_{fdvs} = MV_3^{n-1} \text{ if } \alpha < 0.5 \text{ and } \beta \geq 0.5$$

$$MV_{fdvs} = MV_4^{n-1} \text{ if } \alpha \geq 0.5 \text{ and } \beta \geq 0.5$$

where, as in the Bilinear Interpolation algorithm, α is the ratio between horizontal pixel distance of this reference area from the MV_1^{n-1} and macroblock size, while β is the ratio between vertical pixel distance of this reference area from the MV_1^{n-1} and macroblock size. For example, MV_{fdvs}=MV_2^{n-1} in Figure 4.6.

The same criterion used by FDVS algorithm for choosing the motion vector is presented in [VYLS02]. In this work the choice of the motion vector is formulated in a different way and defined as *majority voting* approach.

This algorithm has a computational complexity lower than that of 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 chosen by FDVS may not be meaningful.

In [SKHK03], the conventional FDVS method is improved to reflect the effect of the macroblock types in the skipped frames.

In [LH03], the Bi-direction Dominant Vector Selection (BDVS) algorithm is presented. It is based on FDVS, but it is designed to re-estimate the dominant motion vectors in video sequences with B frames (decribed in Section 2.4.1) that are not considered in FDVS.

Another algorithm called Generic Bi-directional Dominant Vector Selection (GB-DVS) is proposed in [PK05a]. It is based on FDVS method, and is applicable both to P and B frames.

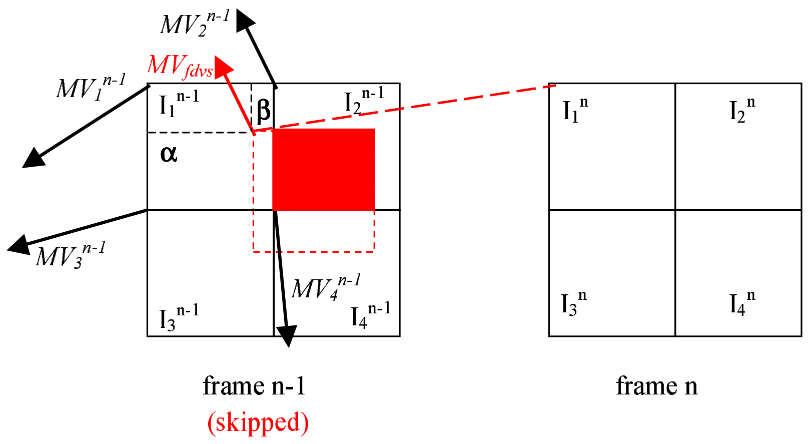


Figure 4.6: Forward Dominant Vector Selection algorithm

4.3.3 Telescopic Vector Composition

A simple algorithm is Telescopic Vector Composition [SG00], that selects, in the skipped frame, the motion vector MV_{tvc} of the macroblock corresponding to the macroblock in the current frame. For example, MV_{tvc}=MV_1^{n-1} in Figure 4.7.

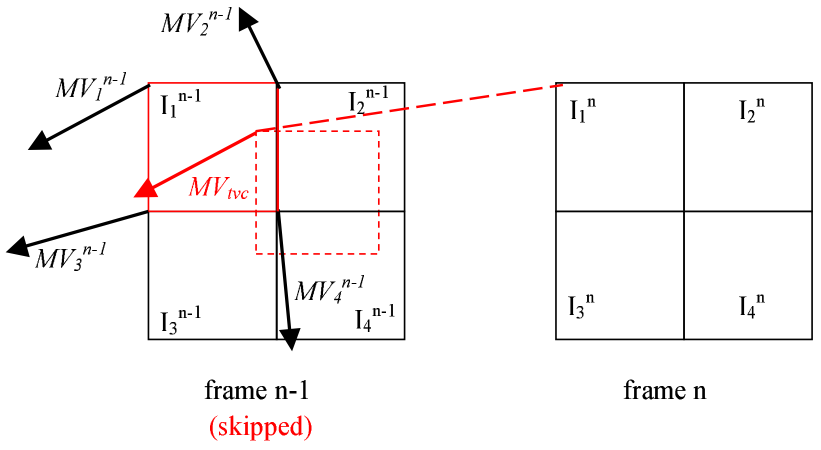


Figure 4.7: Telescopic Vector Composition algorithm

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 for the most part the corresponding macrobloclock in the skipped frame. For this reason, the results obtained by TVC and FDVS can be very close. We will show their performance in the next chapter.

4.3.4 Activity Dominant Vector Selection

The choice of the motion vector is based on the activity of the macroblocks in the Activity Dominant Vector Selection algorithm, presented in [CCP02]. The activity of a macroblock is represented by the number of nonzero quantized DCT coefficients (NZ) of the residual of the blocks belonging to that macroblock. The ADVS algorithm selects the motion vector (MV_{advs}) of the macroblock with the largest activity, among those overlapping 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 4.8, ADVS chooses the motion vector of I_4^{n-1} as dominant vector (MV_{advs}=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 $NZ(I_2^{n-1})$.

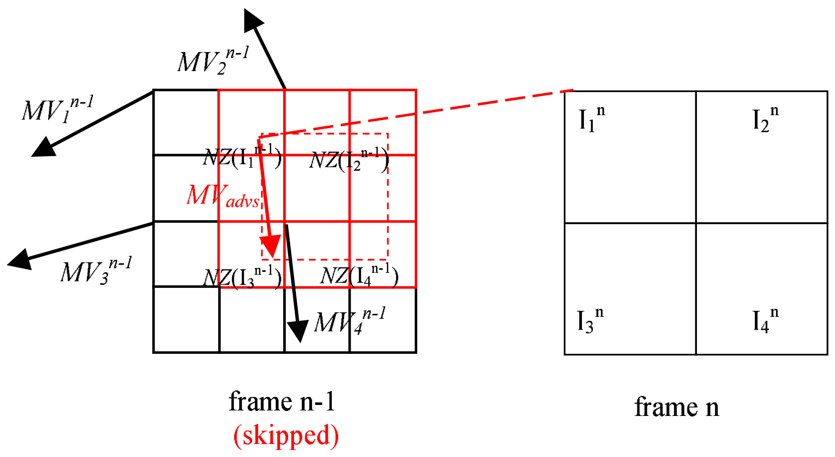


Figure 4.8: Activity Dominant Vector Selection algorithm

The idea of this algorithm is to select the motion vector of the macroblock with maximum activity (NZ) corresponding to larger prediction errors. The larger 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.

4.4 Motion Vector Refinement

Usually, a composed motion vector is not the optimal one, and it should be refined to obtain a better video quality [YSL99][YSL98]. A Full Search motion estimation process can be applied in a new restricted area pointed by the composed motion vector, to find the best possible matching area. A search window of a few pixels is usually adopted: typically, the search range is set to ± 2 pixels around the composed

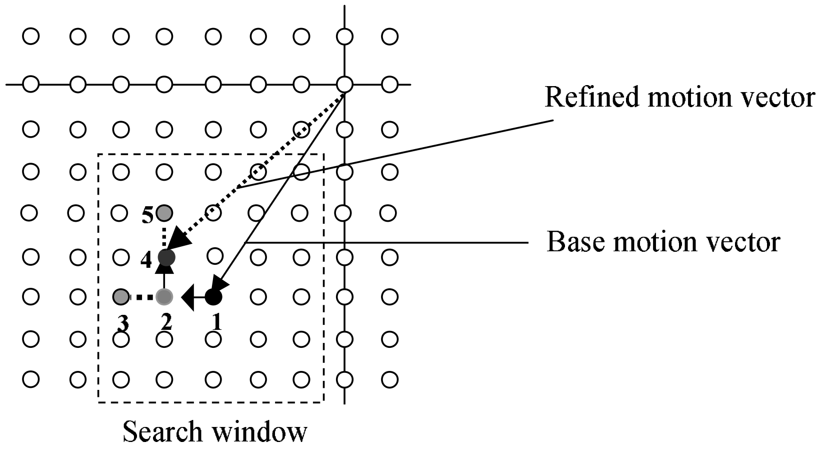


Figure 4.9: Fast Motion Vector Refinement algorithm

motion vector. It has been shown in [SG00] that such a small range can achieve a similar performance to that of the total full-search motion re-estimation.

In [YSL99], a horizontal and vertical fast motion refinement scheme is proposed: instead of analyzing all checking points within the search window, this scheme searches for a minimum point over the horizontal line first, and then over the vertical one, as shown in Figure 4.9, where the best case and a search range of ± 2 pixels are considered.

The SAE (see Equation 2.3) is computed for the starting position in the horizontal search, and only when the computed SAE on the left side is larger than that one of the starting point, the points on the right side are searched. The vertical search is performed in a similar way. In Figure 4.9, the SAE of the starting point at position 1 is compared with that of the adjacent left point at position 2. If the computed SAE at position 2 is smaller, then the point located at position 3 is checked. If the SAE at position 2 is still smaller than that a position 3, a minimum point in the horizontal direction has been found. The search continues in the vertical direction. The SAE of position 2 is compared with that of position 4. If the last one is smaller, it is compared with the SAE of position 5. If the SAE of position 4 is still smaller, a minimum SAE value is found, in both the horizontal and vertical dimensions. Five and seven checking points are required in the best and worst cases, respectively. In Figure 4.9, we report the best case only. In [HWL98], the refinement range is dynamically decided based on the motion vector size and the number of consecutive skipped frames. In [CCP02], an approach similar to that of [YSL99] is pursued, and a variable step-size search algorithm is proposed: the step-size, say S, is computed as function of the composed motion vector components. Then, exactly nine points are checked. Starting from the position pointed by the composed motion vector, the positions S pixels far from it in the horizontal line are checked. Among these three points, the minimum one becomes the starting point for being checked, together

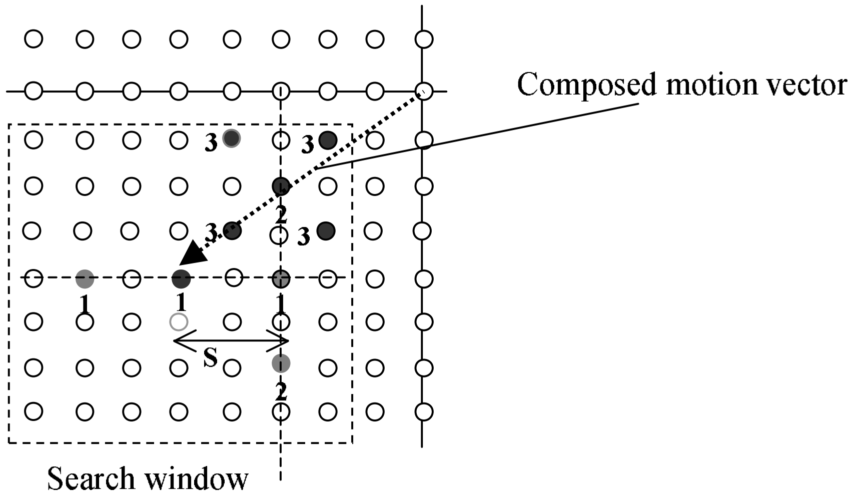


Figure 4.10: Variable step-size Motion Vector Refinement algorithm

with the position S pixels far from it in the vertical line, and the minimum one is chosen. Finally, the four cross positions around it are checked. This scheme with 7x7 search window and S=2 is represented in Figure 4.10.

4.5 Frame skipping policies

The choice of frames to be skipped is an important issue in temporal transcoding. It greatly influences the quality of transcoded video sequences.

Most frame skipping policies are based on motion information to skip not necessary frames. This is why the drawback of temporal transcoding is the jerky effect caused by skipping frames, and it is more evident when frames with large motion are skipped. The basic idea of motion based strategies is that, if the motion information of the frame is larger than a threshold, the frame cannot be skipped since it has considerable motion, and it is not possible to have a good approximation of it by using the previous transcoded frame. It is assumed a communication model where the remote decoder replaces each missing frame with the previous transcoded frame. In Section 4.5.1 we survey the most popular motion based frame skipping policies. We briefly present other frame skipping policies in Section 4.5.2.

4.5.1 Motion-based frame skipping policies

The motion information criterion in frame skipping is studied in depth in this thesis. There are different formulations of the motion information measure. The most popular definition considers how much motion is in a frame. This motion information measure is also called "motion activity". In [HWL98] and [FCS02] it is defined as:

$$MA_n = \sum_{i=1}^{N} |(u_n)_i| + |(v_n)_i| \tag{4.9}$$

where N is the total number of macroblocks in frame n, $(u_n)_i$ and $(v_n)_i$ are the motion vector components of macroblock i in frame n.

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. According to the motion activity, a frame rate control scheme dynamically adjusts the number of skipped frames.

In this thesis, we improve this motion activity based skipping policy as we will show in Section 6.2.

When frames are dropped, re-encoding errors in motion compensated macroblocks cannot be avoided entirely, even if error compensation schemes are applied (as we presented in Section 4.2). In [FCS02], a frame skipping strategy taking into 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 named FSC (Cumulative Frame Skipping) which takes into account the accumulated magnitude of the motion vectors and the re-encoding errors, is defined as:

$$FSC_n(MA_n, RE_{n-1}) = \frac{\sum_{i=1}^{N}(MA_n)_i}{\sum_{i=1}^{N}(RE_{n-1})_i} \tag{4.10}$$

where N is the total number of macroblocks in the current frame n, MA_n is the motion activity of the frame n defined as in (4.9), and RE_{n-1} are the accumulated re-encoding errors due to transcoding, for the motion compensated macroblocks of the current frame as we described in Section 4.2.1 and Section 4.2.2. This metric is compared with a dynamically tuned threshold according to target and outgoing frame rate. A large value of the accumulated re-encoding errors reduces the value of $FSC_n(MA_n, RE_{n-1})$, causing the skipping of the frame.

A different approach that considers the motion activity is presented in [SC04], where a metric representing the motion change is defined. The motion change occurred at the current frame is defined as the difference between the motion of that frame in the transcoded and in the original video sequence. Such motion change is different for skipping and not skipping cases of the previous frame.

As show in Figure 4.11(a), in the original video sequence the motion change that occurred at frame $k+1$ is defined as:

$$\delta mv_{org} = \frac{mv_2 - mv_1}{t} \tag{4.11}$$

where t is the interval time. If frame $k+1$ is trancoded, the motion change that occurred in this frame is the same in the original sequence and in the transcoded sequence. So, no jerky effect is introduced by transcoding. However, when frame

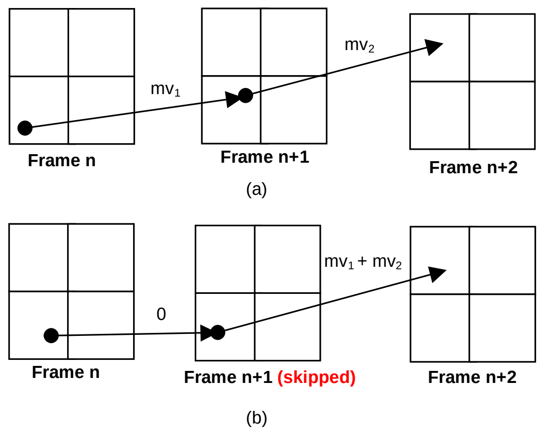


Figure 4.11: Motion based frame skipping: motion change analysis (a) $n+1$ frame is not skipped (b) $n+1$ frame is skipped

$k+1$ is skipped, a direct copy of the previous coded frame k is used for displaying. In this case, the motion change occurred at frame $k+1$ as shown in Figure 4.11(b) can be expressed as:

$$\delta mv_{trans} = \frac{mv_1 + mv_2 - 0}{t} \tag{4.12}$$

If frame $k+1$ is skipped, δmv_{org} and δmv_{trans} are different, and this difference causes an undesired jerky effect. In order to compute this difference, the Square Difference of Motion Change (SDMC) is defined as:

$$SDMC = (\delta mv_{trans} - \delta mv_{org})^2 \tag{4.13}$$

The goal of this policy is to mitigate the difference of motion change between the transcoded and the original video sequence, reducing the jerky effect caused by frame skipping. If $SDMC$ metric is higher than a threshold, it means that a large motion jerky will occur by skipping this frame, so the frame is transcoded. If it is smaller than the threshold, the frame is skipped, since skipping this frame will not introduce visual quality degradation. The threshold is adaptively updated to well reflect the actual motion change of the target video sequence. The proposed frame-skipping control scheme can only be applied for transcoding and not for encoding, since it is needed to know the motion vector of the next frame when transcoding the current frame, and this motion vector is only available in the pre-encoded video.

4.5.2 Other frame skipping policies

A dynamic approach minimizing long runs of consecutive skipped frames is proposed in [CSA03], where a frame skipping control scheme based on a buffer level prediction algorithm is presented. This buffer prediction algorithm defines a prediction temporal window n as the number of future frames over which the transcoding buffer level is estimated. The buffer fullness after transcoding frame $i + n$ is given by:

$$B_{i+n} = B_i + \sum_{j=i+1}^{i+n} (R_j) - n\frac{R_B}{F} \tag{4.14}$$

where B_{i+n} and B_i are the buffer occupancy after transcoding frames $i + n$ and i respectively, R_j is the expected number of bits for frame j, F is frame rate, and R_B is the network transmission rate.

One buffer threshold empirically set to 80% of the buffer size is used to determine whether a future frame will be transcoded, or skipped. After having estimated the buffer level for all frames within the prediction window by using (4.14), the number of consecutive frames to be skipped, immediately after $i + n$, N_{skip}, is determined by:

$$N_{skip} = \frac{B_{i+n} - Threshold \times B_{max}}{R_B/F} \qquad if \quad (B_{i+n} - Threshold \times B_{max}) > 0 \tag{4.15}$$

where B_{max} is the transcoder buffer size. If $N_{skip} > 1$, then several consecutive frames are expected to be skipped. The frame rate control algorithm forces the transcoder to immediately skip the next frame: in this way, the buffer occupancy decreases to the point that one or more future frames will not be skipped as previously estimated. By using this strategy, the number of consecutive skipped frames is reduced. This method does not prevent buffer overflow, so a second threshold, set to 95% of the buffer size, is used for forcing skipping of the next frame if the current buffer level is above it. The critical aspect of this approach is computing the expected number R_j of bits for frame j. To perform this, a log-linear approximation model, described in [CSA02], is used. A drawback of this model is that it is very complex and heavy in the transcoding process.

The buffer fullness and the number of consecutive skipped frames are two important aspects in temporal transcoding, outlined in [BLM05a] and that we will explain in more detail in Section 6.3.2.

In [DKD06], a new approach that takes into account the content information of the video sequence is presented. In this approach, content information is used to skip a frame, according to bandwidth variations and constraints. In particular, if a video frame should be transmitted at time $t = n$, the available network $B(n)$ is estimated at this time t. Let B_0 be the minimum bandwidth required for transmitting a frame in one frame period. If $B(n)$ is smaller than B_0, the ratio $L(n) = B_0/B(n)$, indicating how many times the information should be reduced in order to be transmitted through the network, is computed. The number of frames K that should be skipped is:

$$K = \lceil L(n) \rceil$$

Consequently, the following K frames, namely frames $(n+1, n+2, ..., n+K-1)$, should be skipped to satisfy the current bandwidth requirements. The proposed algorithm, among the current and the candidate frames for skipping $(n, n+1, n+2, ..., n+K-1)$, selects the most representative frame to be delivered, whereas the remaining frames are skipped. A measure, called "feature vector", taking into account the colour information and the motion information in a frame, is used to select the most representative frame. Let f_i be the feature vector of a frame i. Among the K candidate frames, the average feature vector f over all K frames is computed in this way:

$$\bar{f} = \frac{1}{K} \sum_{i=1}^{K} (f_i) \tag{4.16}$$

The most representative frame is frame J, whose feature vector agrees with:

$$J = arg \min_{i} \| f_i - \bar{f} \| \tag{4.17}$$

The network bandwidth is re-estimated after the completion time of each representative frame transmission. The authors of [DKD06] propose also an objective evaluation scheme different from conventional *PSNR*. It is based on the "feature vector" measure and it is used to evaluate the proposed approach.

The idea of selecting the most representative frame to be transmitted, when a number of frames has to be skipped, in order to meet the bandwidth constraints was proposed also in [BLM05a], for the case of constant bit rate reduction. We will address this problem in depth in Chapter 6.

Part II

Proposed Approaches

Chapter 5

Temporal Transcoding: architecture and Motion Vector Computation

Abstract

In this chapter, we address temporal video transcoding for real-time communication. We describe the adopted pixel domain transcoding architecture and the followed approaches to perform motion vector computation. An H.264 compliant motion vector composition scheme is proposed.

5.1 Introduction

Temporal transcoding is the focus of this thesis. Our goal is to address the issues of temporal transcoding for improving real-time video communication of many multimedia services (video telephony, video conference, etc), on mobile networks. For this purpose, the proposed solutions try to optimize the transcoding performance, keeping a good quality of transcoded video sequences. In this chapter, we approach two main problems of temporal transcoding: transcoder architecture design and motion vector computation. The adopted frame skipping strategies will be presented in the next chapter.

Our transcoding architecture is described in Section 5.2. To reduce transcoding complexity, we adopt motion vector composition for re-computing motion vectors, when a frame is skipped. In Section 5.4 we present a performance comparison of different motion vector composition algorithms implemented in our transcoding architecture.

Finally, the problem of motion vector composition in H.264 based transcoder is addressed. In Section 5.5, we propose a new motion vector composition scheme

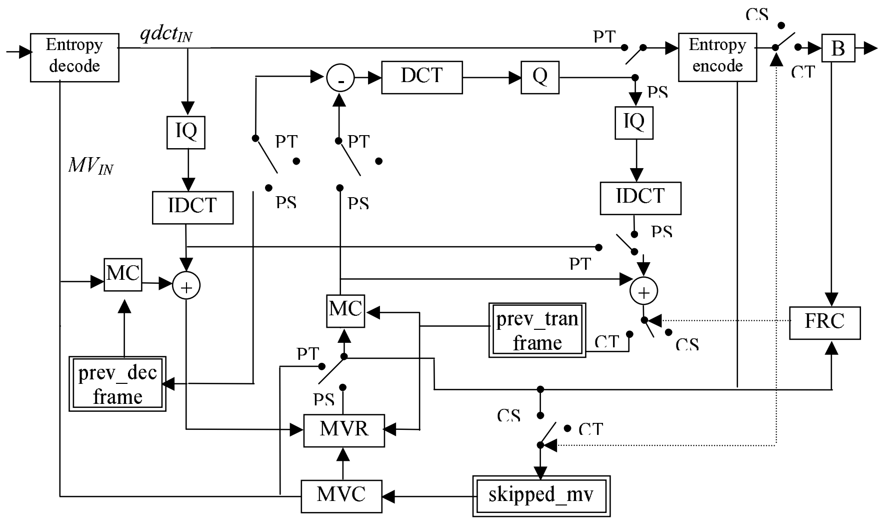


Figure 5.1: Our temporal transcoder architecture

to deal with variable partitioning of H.264 coded frames. This result is one of the original contributions of this thesis, published in [LM07a].

5.2 Transcoder Architecture

We developed a pixel-domain temporal transcoder architecture. In our transcoder architecture, the words "input" and "output" are always related to the transcoder, and we call IR the input bit rate, and R the output bit rate. Such architecture is able to reduce the input bit rate IR of the incoming video sequence, by eliminating some frames, so that the output bit rate R turns out to be constant as we will explain in Chapter 6. In this case, the frame rate of the output video sequence is not constant, and we assumed that the skipped frames are replaced by the previous ones (freezing) at displaying time in the final decoder. In our transcoder, the motion vectors are computed by choosing one of the four motion vector composition algorithms, (Bilinear Interpolation [HWL98], Forward Dominant Vector Selection [YSL99], Telescopic Vector Composition [SG00] and Activity Dominant Vector Selection [CCP02]), described in Section 4.3, and a motion vector refinement procedure (see Section 4.4). The residual is computed in the pixel domain. A re-estimation procedure of the type of macroblocks of a frame when its reference frame is skipped is developed. In addition, an intra-refresh procedure, to correct re-encoding errors introduced by skipping of frames, is introduced. Both procedures are explained in more detail in Section 5.3. In this transcoder architecture, the main parts the we investigated are Motion vector composition (MVC) and Frame Rate Control (FRC). About MVC a performance comparison of motion vector compositions algorithms is performed and a new motion vector composition scheme for H.264 transcoder is proposed. About FRC different frame skipping strategies are proposed.

The architecture of our transcoder is shown in Figure 5.1. The behavior of the transcoder is different according to the decision taken for 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 and transmitted. The switching between the two behaviors is represented in Figure 5.1 by the switch PS/PT. In addition, also transcoding or skipping of the current frame determines a different behavior of the transcoder. The switching between the two behaviors is represented in Figure 5.1 by the switch CS/CT. Every incoming frame in_frame is decoded in the transcoder's decoder, by means of motion compensation with the previous decoded frame $prev\_dec\_frame$.

In Section 5.2.1 and Section 5.2.2, we present the configuration of the transcoder architecture, when the transcoded reference frame has been transmitted and skipped, respectively. Note that, in this architecture we use the term *transcoded* to indicate that a frame is coded with respect to the last transcoded and transmitted frame. So, each frame is transcoded before deciding if it is better to skip or transmit it. This is because it is needed to know some features of the transcoded frame (for instance, its size in bits or motion information), before applying some frame skipping policies. Transcoding each frame is needed also for avoiding to store all skipped frames between the current frame and the last transmitted one, which implies large memory resources. Transcoded frames are then skipped or placed in the buffer for being transmitted.

5.2.1 Transcoder architecture: reference frame not skipped

When the reference frame has been transcoded and transmitted (Figure 5.2), the current frame is decoded by inverse quantization (IQ), inverse DCT (IDCT), and motion compensated (MC) with respect to the last decoded frame, to update $prev\_dec\_frame$. The motion compensation is performed with respect to the last transcoded and transmitted frame to update $prev\_tran\_frame$, in case that the Frame Rate Control (FRC) module decides to transcode and transmit the current frame. In this case, the motion vectors (MV_{IN}) and DCT coefficients ($qdct_{IN}$) are re-encoded and put in the output buffer. 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 to perform Motion Vector Composition (MVC) for the next incoming frame.

5.2.2 Transcoder architecture: reference frame skipped

In case of skipped reference frame (Figure 5.3), it is needed to recompute the motion vectors and the residual of the current frame with respect to the last transcoded and transmitted frame. The motion vectors are computed by means of Motion

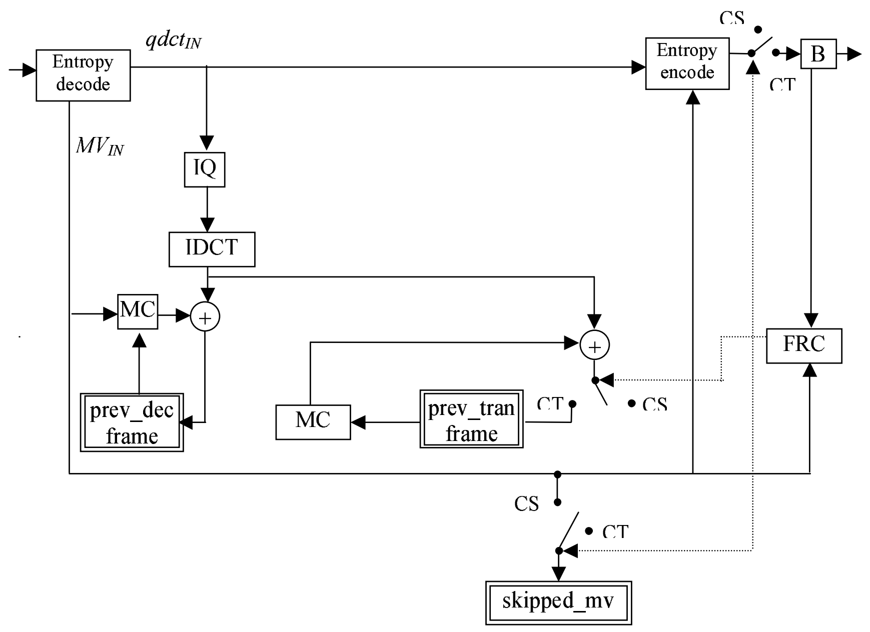


Figure 5.2: Our temporal transcoder architecture: not skipped reference frame

Vector Composition (MVC module in Figures 5.1, 5.3) and Motion Vector Refinement (MVR module in Figures 5.1, 5.3). The MVC module adds to the vectors of the incoming frame MV_{IN} the motion vectors chosen among *skipped_mv*, by one of the implemented motion vector composition algorithms. As in the case of not skipped reference frame, the current frame is decoded by inverse quantization,

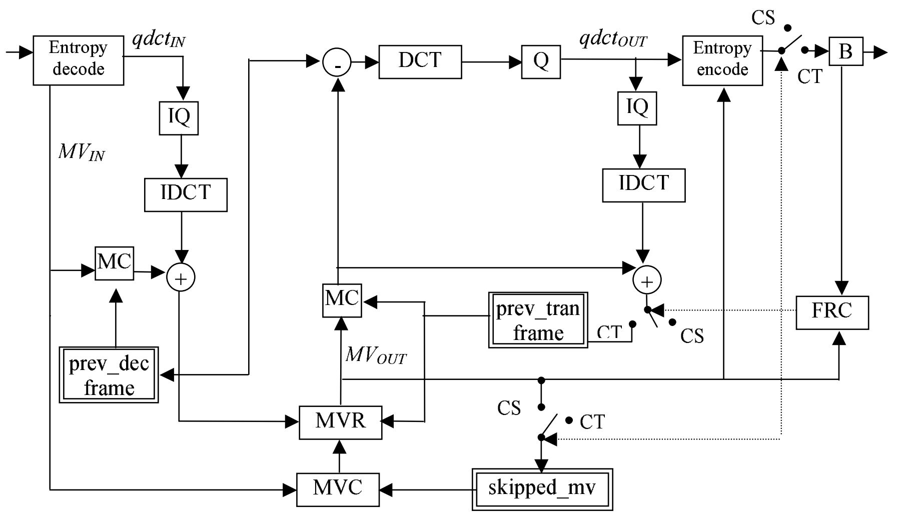


Figure 5.3: Our temporal transcoder architecture: skipped reference frame

inverse DCT, and motion compensated with respect to the last decoded frame, to update $prev\_dec\_frame$. The motion compensation is performed also with respect to the last transcoded and transmitted frame $prev\_tran\_frame$ to produce the motion-compensated frame. This one, together with the current decoded frame and the motion vectors obtained by MVC module, are the inputs for MVR module. The MVR module performs a refined motion estimation around the vectors given by MVC, in order to produce the best possible motion vectors. It can be performed with Full Search Motion Estimation (FSME) or Fast Motion Refinement Algorithm (MEFast). These motion vectors (MV_{OUT}) are used to perform motion compensation with respect to the last not skipped frame $prev\_tran\_frame$. The obtained motion-compensated frame is then subtracted from the current decoded frame to produce the residual for the current frame. The residual is coded by DCT and quantization to obtain the quantized DCT coefficients $qdct_{OUT}$. Transcoded frames are then skipped or placed in the buffer for being transmitted. If the FRC module decides to transmit the transcoded current frame, the quantized DCT coefficients $qdct_{OUT}$ and the motion vectors MV_{OUT} are entropy coded. In addition, the quantized DCT coefficients, after inverse DCT and quantization, will be added to the motion compensated frame, in order to store the current frame in $prev\_tran\_frame$. Otherwise, if the transcoded frame will be skipped, MV_{OUT} will be stored in *skipped_mv*, for being used to perform Motion Vector Composition (MVC) for the next incoming frame.

5.3 MVC in our transcoder

In our temporal transcoder, when the reference frame has been skipped, we use Motion Vector Composition (MVC), (surveyed in Section 4.3), to obtain the motion vectors for the current frame. The main issue in temporal transcoding is that, the motion vectors and the type of macroblocks of the current frame are no more valid, when the reference frame is skipped, since they refer to a frame that is not in the output video sequence at transcoder. The implemented motion vector composition module re-estimates the type and motion vectors of macroblocks of the current frame when its reference frame is skipped, as follows:

- If a macroblock was coded as SKIPPED, it must be coded in the transcoder with the same type of the corresponding macroblock in the skipped frame. If the corresponding macroblock in the skipped frame is an INTER macroblock, its motion vectors become the motion vectors of the macroblock in the current frame. Later, in the transcoding process, also the residual of this macroblock in the skipped frame will be the residual of macroblock in the current frame. This is why, in the front decoder, a SKIPPED macroblock is usually replaced with the corresponding macroblock in the previous frame. In this case, the previous frame is the skipped frame;

- If a macroblock was coded as INTRA, it must be coded as INTRA macroblock again, with motion vector equal to zero;
- If a macroblock was coded as INTER, and it is a non MC (motion compensated) macroblock, it must be coded with the type and motion vector of the corresponding macroblock in the skipped frame, as follows:
  - If the corresponding macroblock was coded as INTRA in the skipped frame, the type of the macroblock in the current frame will be INTRA, and its motion vector will be equal to zero;
  - If the corresponding macroblock was coded as INTER in the skipped frame, the type of the macroblock in the current frame will be INTER as well, and its motion vector will be equal to that of the macroblock in the skipped frame;
  - If the corresponding macroblock was coded as SKIPPED in the skipped frame, the type of the macroblock in the current frame will be SKIPPED, and its motion vector will be equal to that one of the macroblock in the skipped frame.
- If a macroblock was coded as INTER, and it is an MC (motion compensated) macroblock, a motion vector composition algorithm can be applied to find the motion vector of the reference area in the skipped frame pointed by the the motion vector of the current macroblock. In our transcoder we implemented four motion vector composition algorithms, namely Bilinear Interpolation [HWL98], Forward Dominant Vector Selection [YSL99], Telescopic Vector Composition [SG00] and Activity Dominant Vector Selection [CCP02] (described in Section 4.3). The motion vector can be computed by choosing one of these motion vector composition algorithms (specified in the configuration file of transcoder) and it is added to the motion vector of the current macroblock, obtaining the re-estimated motion vector of the current macroblock with respect to the last not skipped frame as explained in Section 4.3.

We present the pseudo-code of our MVC module in Table 5.1, where MV_{IN} are the motion vectors of the current frame, MV_{SKIP} are the motion vectors of macroblocks in the skipped frame, MV_{OUT} are the re-estimated motion vectors of the current frame after MVC, $Type_{IN}$ is the type of the current macroblock, $Type_{SKIP}$ is the type of the corresponding macroblock in the skipped frame, $Type_{OUT}$ is the re-estimated type of the current macroblock after MVC, MVC_{ALG} is the chosen motion vector composition algorithm.

Note that, this MVC module description is referred to our MVC implementation considering an MPEG4 based transcoder, where each INTER macroblock (16×16 pixels) can be partitioned in four blocks (8×8 pixels). So, we can have four motion vectors for each INTER macroblock. We call this macroblock INTER4v.

Table 5.1: Pseudo-code of MVC module

```
Input = $MV_{IN}$, $MV_{SKIP}$, $Type_{IN}$, $Type_{SKIP}$, $MVC_{ALG}$
Output = $MV_{OUT}$, $Type_{OUT}$
 1. for all (MB in CurrentFrame) do
 2.   if $Type_{IN}[MB]$ is SKIPPED then
 3.     $MV_{OUT}[MB]$=$MV_{SKIP}[MB]$
 4.     $Type_{OUT}[MB]$=$Type_{SKIP}[MB]$
 5.   else if $Type_{IN}[MB]$ is INTRA then
 6.     $MV_{OUT}[MB]$=(0, 0)
 7.     $Type_{OUT}[MB]$=INTRA
 8.   else ($Type_{IN}[MB]$ is INTER or INTER4v)
 9.     if$MV_{IN}[MB]$=(0,0)
10.       if$Type_{SKIP}[MB]$ is INTRA then
11.         $Type_{OUT}[MB]$=INTRA
12.       if$Type_{SKIP}[MB]$ is SKIPPED then
13.     $Type_{OUT}[MB]$=SKIPPED
14.       if$Type_{SKIP}[MB]$ is INTER then
15.     $Type_{OUT}[MB]$=INTER
16.     $MV_{OUT}[MB]$=$MV_{SKIP}[MB]$
17.     else
18.       $Type_{OUT}[MB]$= INTER
19.       if$MVC_{ALG}$ is BI then
20.         $MV_{OUT}[MB]$=$MV_{IN}[MB]$ + BI($MV_{SKIP}[MB]$)
21.       if$MVC_{ALG}$ is TVC then
22.         $MV_{OUT}[MB]$=$MV_{IN}[MB]$ + TVC($MV_{SKIP}[MB]$)
23.       if$MVC_{ALG}$ is FDVS then
24.         $MV_{OUT}[MB]$=$MV_{IN}[MB]$ + FDVS($MV_{SKIP}[MB]$)
25.       if$MVC_{ALG}$ is ADVS then
26.         $MV_{OUT}[MB]$=$MV_{IN}[MB]$ + ADVS($MV_{SKIP}[MB]$)
27.       end if
28.     end if
29.   end if
30. end for
```

We apply MVC for each block of an INTER4v macroblock, as described before, for the case of non motion compensated and motion compensated INTER macroblocks. Having the real-time communication as target application, we assume that there are not *bidirectional* frames in the transcoder input video sequence, to reduce the complexity of coding process.

The problem of motion vector composition in H.264 based transcoder is addressed in Section 5.5, where a MVC scheme is proposed, taking into account the variable macroblock partitioning of coded frames.

As in [VYLS02], we adopt an intra refresh technique able to correct errors introduced in transcoding architecture, due to skipping of frames (see section 4.2.1). In the proposed intra refresh technique, a macroblock is coded as INTRA when its residual is greater than a properly tuned threshold. Such threshold is dynamically computed according to the average residual of the previous frames in a sliding window. However, a macroblock is coded as INTRA, when the amount of bits needed to code its residual is equal to the amount needed to code it as an INTRA macroblock.

5.4 MVC algorithms comparison

In this section we want to present a performance comparison of motion vector composition algorithms implemented in our transcoding architecture (see Section 5.3), in terms of PSNR of transcoded video sequence and computation time of the entire transcoding process.

We evaluate the PSNR (see Section 2.5), indicating the quality of a video sequence, by taking into account the differences of luminance values of corresponding pixels in the original and reconstructed frames of the video sequence. We compute the PSNR in this way: we consider as original video sequence the one decoded after the front encoder. As reconstructed sequence, we use the one decoded after our transcoder, where skipped frames are replaced with their previous ones (freezing). This way of computing the PSNR allows us to measure the actual visual quality perceived by the final user, considering the degradation introduced by the transcoding process only, without considering those introduced by the previous coding process. For the sake of clearness, in the following of this thesis, we will call this measure PSNR1. A more detailed analysis of used PSNR measures will be presented in Section 6.3.1.

We show the average PSNR1 achieved with the four implemented motion vector composition algorithms: Bilinear Interpolation (BI), Forward Dominant Vector Selection (FDVS), Telescopic Vector Composition (TVC) and Activity Dominant Vector Selection (ADVS). Their performance is evaluated with respect to that of Full Search Motion Estimation (FSME).

In this thesis we use for simulations, well known benchmark video sequences with 300 frames, QCIF format, and 30 fps [vid]. They have different motion features: “akiyo” and “mobile” have little motion, “foreman”, “carphone”, ‘table” and “coastguard” have motion and scene change.

After Motion Vector Composition, a motion vector refinement of ±2 pixels around the composed motion vector is applied. The search method that has been used in these experiments is the Full Search method with Sum of Absolute Errors (SAE) matching criterion (see Section 2.3.1).

We evaluate the average PSNR1 for different input bit rates of the transcoder, and two frame rate reductions (30 to 15 fps and 30 to 7.5 fps). In this way, we can estimate the performance of all MVC algorithms when the same frames are skipped. Note that, to obtain a constant frame rate reduction, the transcoder reduces the frame rate without using any buffer control and without operating on quantization parameters, so the transcoder output bit rate is not constant. The presented results are related to a temporal transcoder that we implemented by using the MPEG4 reference software Visual Simple profile [ISO04].

In Table 5.2 and Table 5.3 we show PSNR1 of "foreman", "akiyo", "coastguard" and "carphone" for 30 to 15 fps and 30 to 7.5 fps respectively. We can observe that in both frame rate reduction cases, all motion vector composition algorithms show a PSNR1 close to that one of FSME. BI, FDVS and ADVS have a better performance than TVC. This is showed also in Figures 5.4 and 5.5.

In Table 5.4 and Table 5.5 we show the computational complexity of the entire transcoding process with FSME and MVC algorithms. We used an Athlon 64 3200+, with 512MB memory and Linux Mandrake 10.2 as operating system, to run our experiments. The presented values relate to the ratio of computation time of the implemented motion vector composition algorithms with respect to FSME. The reported values are the average values obtained by running the same experiment 20 times. In Figure 5.6 we show this computation time ratio for "foreman" video sequence by averaging the computation times of all considered input bit-rates. All motion vector composition algorithms reduce the computation time of the transcoding process and this reduction is higher for video sequences with much motion and for high frame rate reductions (30 to 7.5 fps) because in these cases the FSME process is more complex. In most cases, all motion vector composition algorithms achieve a reduction of the computation time of the transcoding process of more than 50%.

5.5 MVC in H.264 transcoding

In this section we propose a MVC algorithm for H.264 based transcoding. After describing the particular features of this new video coding standard we address the problem of motion vector computation in H.264 temporal transcoding. Then, we present our contribution to this research area and the performance results of the proposed algorithm.

5.5.1 Motivations

MPEG-4/AVC/H.264 (H.264 for short) is a video coding standard, born by the joint efforts of ITU-T Video Coding Experts Group (H.263) and ISO/IEC Motion

Table 5.2: MVC algorithms performance comparison: PSNR1 of different video sequences at different input bit rate IR(Kbps) from 30 to 15 fps

Algorithm	IR=32	IR=64	IR=128	IR=256	IR=512
foreman					
FSME	30.3	30.55	30.89	32.68	34.30
BI	29.54	30.10	30.75	32.2	34.11
TVC	29.33	29.92	30.61	32.00	34.11
FDVS	29.81	30.20	30.62	32.49	34.11
ADVS	29.99	30.34	30.66	32.35	34.11
akiyo					
FSME	35.49	38.66	42.70	42.81	45.90
BI	35.45	38.63	42.70	42.80	45.90
TVC	35.46	38.64	42.70	42.80	45.90
FDVS	35.46	38.65	42.70	42.80	45.90
ADVS	35.41	38.61	42.69	42.80	45.89
coastguard					
FSME	30.06	29.62	31.12	32.28	33.49
BI	29.78	29.50	31.09	32.26	33.47
TVC	29.37	29.54	31.11	32.27	33.47
FDVS	29.67	29.52	31.12	32.26	33.47
ADVS	29.52	29.46	31.07	32.25	33.46
carphone					
FSME	31.34	29.90	35.64	37.19	37.59
BI	30.91	29.82	35.64	37.18	37.59
TVC	30.81	29.82	35.62	37.18	37.59
FDVS	30.94	29.83	35.64	37.18	37.59
ADVS	30.37	29.78	35.54	37.11	37.58

Table 5.3: MVC algorithms performance comparison: PSNR1 of different video sequences at different input bit rate *IR*(Kbps) from 30 to 7.5 fps

Algorithm	IR=32	IR=64	IR=128	IR=256	IR=512
foreman					
FSME	27.25	28.19	28.5	29.56	30.69
BI	26.7	27.47	27.69	28.4	29.66
TVC	26.6	27.29	27.55	28.51	29.42
FDVS	26.67	27.62	27.86	28.66	29.96
ADVS	27.00	27.79	27.98	28.78	30.09
akiyo					
FSME	34.63	37.36	40.99	41.03	43.38
BI	34.62	37.33	41.00	41.02	43.37
TVC	34.61	37.32	40.99	41.02	43.37
FDVS	34.60	37.34	40.99	41.02	43.37
ADVS	34.60	37.31	41.01	39.13	43.36
coastguard					
FSME	28.51	28.06	28.76	29.44	30.19
BI	28.29	27.69	28.51	29.41	30.17
TVC	28.21	27.67	28.50	29.40	30.16
FDVS	28.20	27.88	28.63	29.42	30.16
ADVS	27.86	27.68	28.51	29.41	30.15
carphone					
FSME	30.09	30.06	32.75	33.88	34.09
BI	28.93	29.36	32.28	33.79	33.86
TVC	27.80	29.79	32.00	32.25	33.16
FDVS	28.16	29.71	31.82	32.45	33.99
ADVS	29.25	29.65	32.55	32.97	34.00

Table 5.4: MVC algorithms performance comparison: Complexity ratio of different video sequences at different input bit rate IR(Kbps) from 30 to 15 fps

Algorithm	IR=32	IR=64	IR=128	IR=256	IR=512
foreman					
FSME	1	1	1	1	1
BI	0.41	0.38	0.36	0.38	0.39
TVC	0.41	0.38	0.37	0.37	0.39
FDVS	0.42	0.38	0.37	0.38	0.40
ADVS	0.41	0.39	0.36	0.38	0.39
akiyo					
FSME	1	1	1	1	1
BI	0.78	0.82	0.85	0.88	0.91
TVC	0.79	0.82	0.84	0.87	0.91
FDVS	0.78	0.83	0.84	0.88	0.90
ADVS	0.79	0.82	0.85	0.88	0.91
coastguard					
FSME	1	1	1	1	1
BI	0.37	0.34	0.33	0.32	0.31
TVC	0.38	0.35	0.33	0.33	0.30
FDVS	0.38	0.34	0.33	0.33	0.30
ADVS	0.38	0.35	0.34	0.33	0.32
carphone					
FSME	1	1	1	1	1
BI	0.44	0.43	0.43	0.45	0.46
TVC	0.43	0.42	0.43	0.44	0.45
FDVS	0.44	0.42	0.44	0.45	0.46
ADVS	0.44	0.42	0.43	0.46	0.45

Table 5.5: MVC algorithms performance comparison: Complexity ratio of different video sequences at different input bit rate IR(Kbps) from 30 to 7.5 fps

Algorithm	IR=32	IR=64	IR=128	IR=256	IR=512
foreman					
FSME	1	1	1	1	1
BI	0.33	0.34	0.33	0.33	0.35
TVC	0.32	0.32	0.32	0.33	0.34
FDVS	0.34	0.34	0.33	0.34	0.35
ADVS	0.33	0.33	0.34	0.33	0.35
akiyo					
FSME	1	1	1	1	1
BI	0.75	0.80	0.83	0.85	0.87
TVC	0.76	0.79	0.82	0.84	0.87
FDVS	0.75	0.80	0.83	0.85	0.88
ADVS	0.76	0.80	0.83	0.85	0.87
coastguard					
FSME	1	1	1	1	1
BI	0.35	0.31	0.30	0.30	0.20
TVC	0.34	0.31	0.31	0.29	0.20
FDVS	0.35	0.32	0.31	0.30	0.28
ADVS	0.34	0.32	0.32	0.30	0.29
carphone					
FSME	1	1	1	1	1
BI	0.41	0.39	0.40	0.41	0.42
TVC	0.41	0.39	0.39	0.40	0.41
FDVS	0.42	0.40	0.40	0.41	0.42
ADVS	0.42	0.40	0.40	0.41	0.41

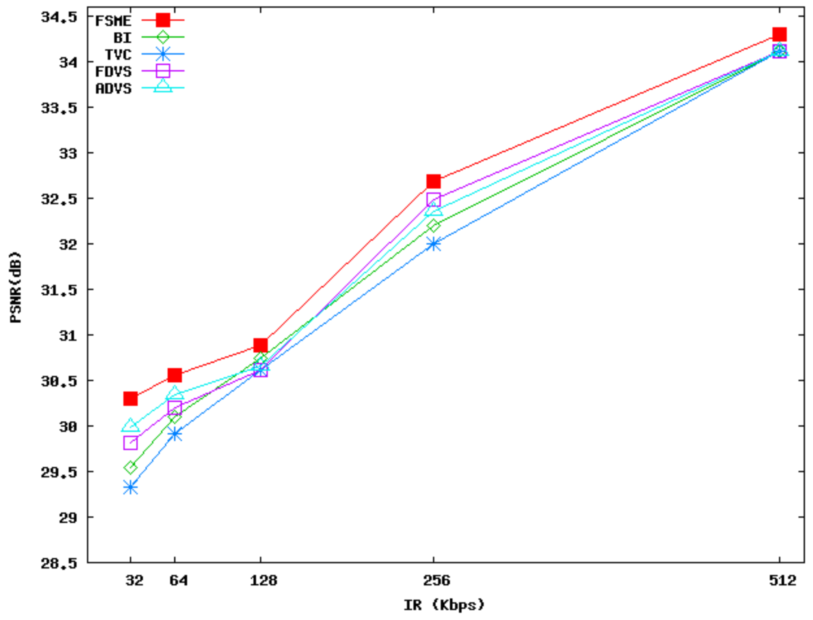


Figure 5.4: MVC algorithms comparison: PSNR1 of "foreman" video sequence (from 30 to 15 fps)

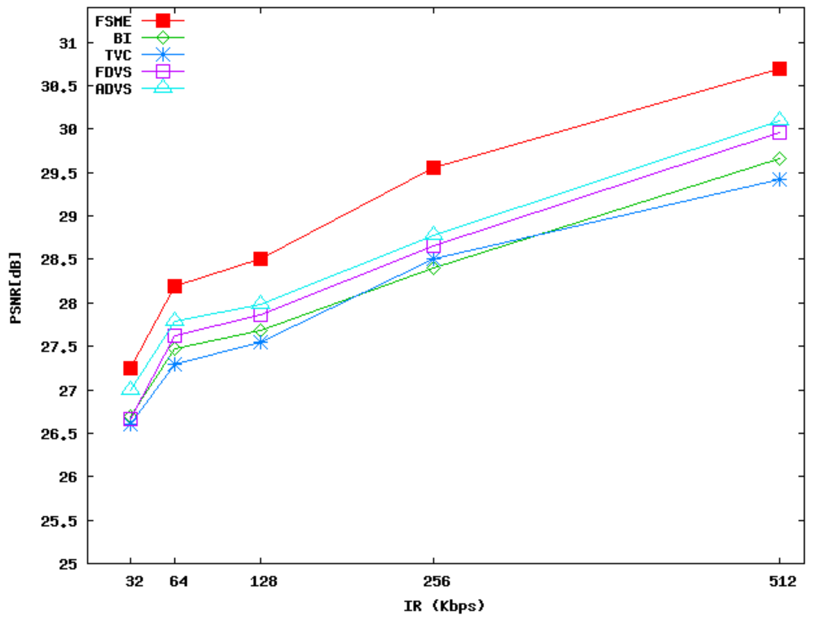


Figure 5.5: MVC algorithms comparison: PSNR1 of "foreman" video sequence (from 30 to 7.5 fps)

Picture Expert Group (MPEG), partnership known as the Joint Video Team (JVT). H.264 codec has been developed to enhance compression performance and to ease

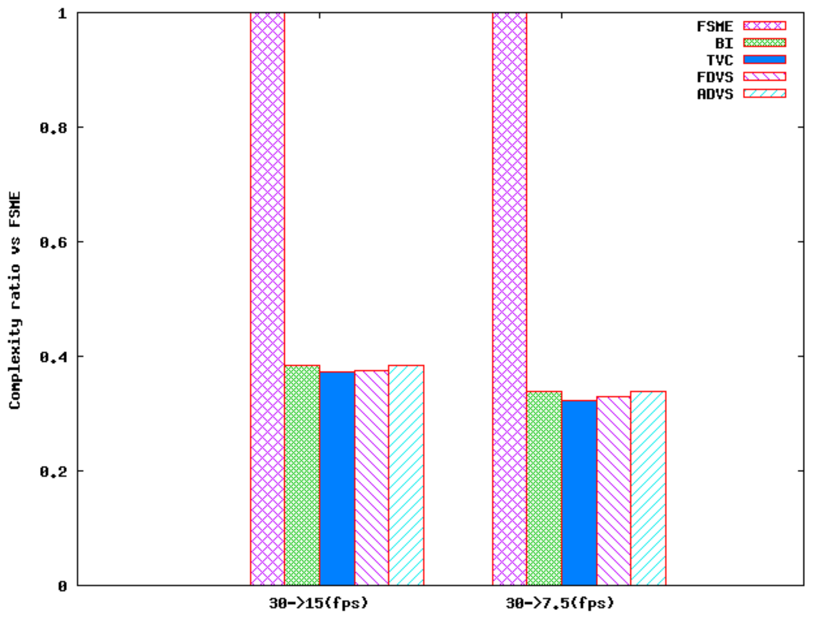


Figure 5.6: MVC algorithms comparison: complexity ratio of "foreman" video sequence

network interaction. The former objective has been reached by defining a Video Coding Layer (VCL), designed to efficiently represent the video content, while the latter, by inserting a Network Abstraction Layer (NAL), which is responsible for packetization and adaptation of the video content to the underlying network or storage media. Relative to prior video coding methods, there are some features of the design that enable enhanced coding efficiency [WSBL03]: quarter-sample accurate motion compensation, motion vectors over picture boundaries, multiple reference picture motion compensation, deblocking filtering, arithmetic entropy coding, and context-adaptive entropy coding.

An important feature of the H.264 codec, which is of interest in this thesis, is the variable block-size partitioning of frames. In particular, the luminance component of each macroblock (16 × 16 samples) can be split in four ways (Figure 5.7), and motion compensated either as one 16 × 16 macroblock partition or two 16 × 8 or 8 × 16 partitions, or four 8 × 8 partitions. If the 8 × 8 mode is chosen, each of the four 8 × 8 sub-macroblocks within the macroblock, may be further split in four ways, either as one 8 × 8 sub-macroblock partition or two 8 × 4 or 4 × 8 sub-macroblock partitions, or four 4 × 4 sub-macroblock partitions [I.E03].

We took into account this variable block-size partitioning of frames in our H.264 temporal transcoder development. Considering an architecture very similar to that one shown in Section 5.2, when a frame is skipped, it is not easy to estimate optimum motion vectors and block types for the current macroblock by using those ones of the skipped frame, since for each macroblock overlapping the reference area in

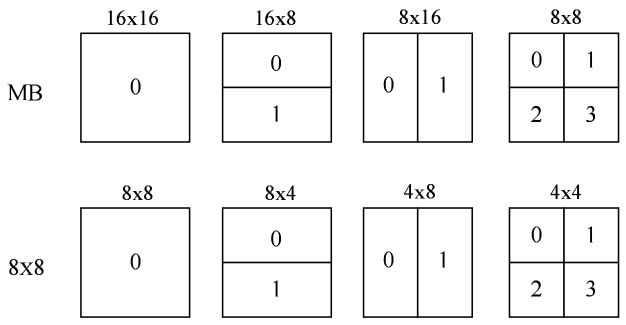


Figure 5.7: Partition modes of a macroblock in H.264/AVC

the skipped frame, there are many motion vectors corresponding to its different partitions.

The optimized motion vector can be obtained by re-estimating a new motion vector and a block type in the transcoder. However, motion estimation requires high computational complexity. Fast motion estimation schemes and mode decision for H.264 bit-rate reduction are proposed in [LTYB05].

In [SLP04], a block adaptive motion vector re-sampling method for H.264 transcoding is proposed. It estimates the motion vectors of all 4×4 sub-blocks of the current $M \times N$ macroblock by taking into account the areas of the overlapped blocks in the skipped reference frame, and the motion vectors of these blocks. The obtained motion vectors for the sub-blocks are used to compute the motion vector of the current $M \times N$ macroblock. The transcoded block types are those ones of incoming blocks when quantization parameters are not updated in transcoding. In addition, a RDO (Rate Distortion Optimization) method is combined with the above motion vector re-sampling method in order to select the most optimized block type in terms of rate distortion performance, when quantization parameters in the transcoder are different from those of the original compressed bitstream.

Due to the variable block-size partitioning described above, the existing MVC algorithms [HWL98][YSL99][SG00][CCP02] cannot be applied without changes, since a separate motion vector is required for each partition or sub-macroblock. We approach this problem and we present our solution in the next section.

5.5.2 Multi-level MVC algorithm

For taking into account the variable partitioning of H.264 frames, we propose a new motion vector composition algorithm adopting a multi-level motion vector composition scheme, together with the Bilinear Interpolation function, that we call Multi-Level Bilinear Scheme (MLBS) [LM07a]. We choose Bilinear Interpolation algorithm because with respect to the other motion vector composition algorithms (FDVS, ADVS, TVC), it takes into account all motion vectors of the macroblocks

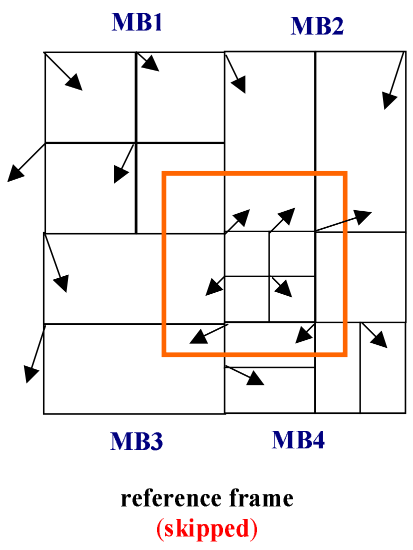


Figure 5.8: H.264 temporal transcoding: motion vectors of partitions overlapping the reference area in the skipped frame

overlapping the reference area in the skipped frame by composing them by Bilinear Interpolation function. In addition, it has a PSNR1 higher than TVC and a time complexity lower than FDVS and ADVS algorithms as we showed in Section 5.4.

The goal of motion vector composition procedure is to find a motion vector for the reference area pointed by the current motion vector, in the last skipped frame. The obtained motion vector is composed with the motion vector of the current macroblock, in order to obtain a motion vector for the current macroblock pointing to the last not skipped frame. Bilinear Interpolation algorithm composes the motion vectors of four macroblocks overlapping the reference area in the skipped frame, according to Equation 4.8 (see Section 4.3.1).

In H.264 codec, for each macroblock overlapping the reference area in the skipped frame, there are many motion vectors (at most 16 motion vectors) corresponding to different partitions of such macroblock, (see Figure 5.8). So, composing all motion vectors of all macroblocks partitions overlapping the reference area in the skipped frame is needed.

We adopted a multi-level scheme to compose these vectors according to Bilinear Interpolation function (Equation 4.8). In Figure 5.9 the behaviour of the proposed scheme is shown for the partitioning of the reference frame presented in Figure 5.8. At the first level, we consider the skipped reference frame, and its macroblocks overlapping the reference area pointed by the current motion vector. The number of macroblocks overlapping the reference area can vary from zero to four. For each of these macroblocks, its partitions are considered at second level, in the following way:

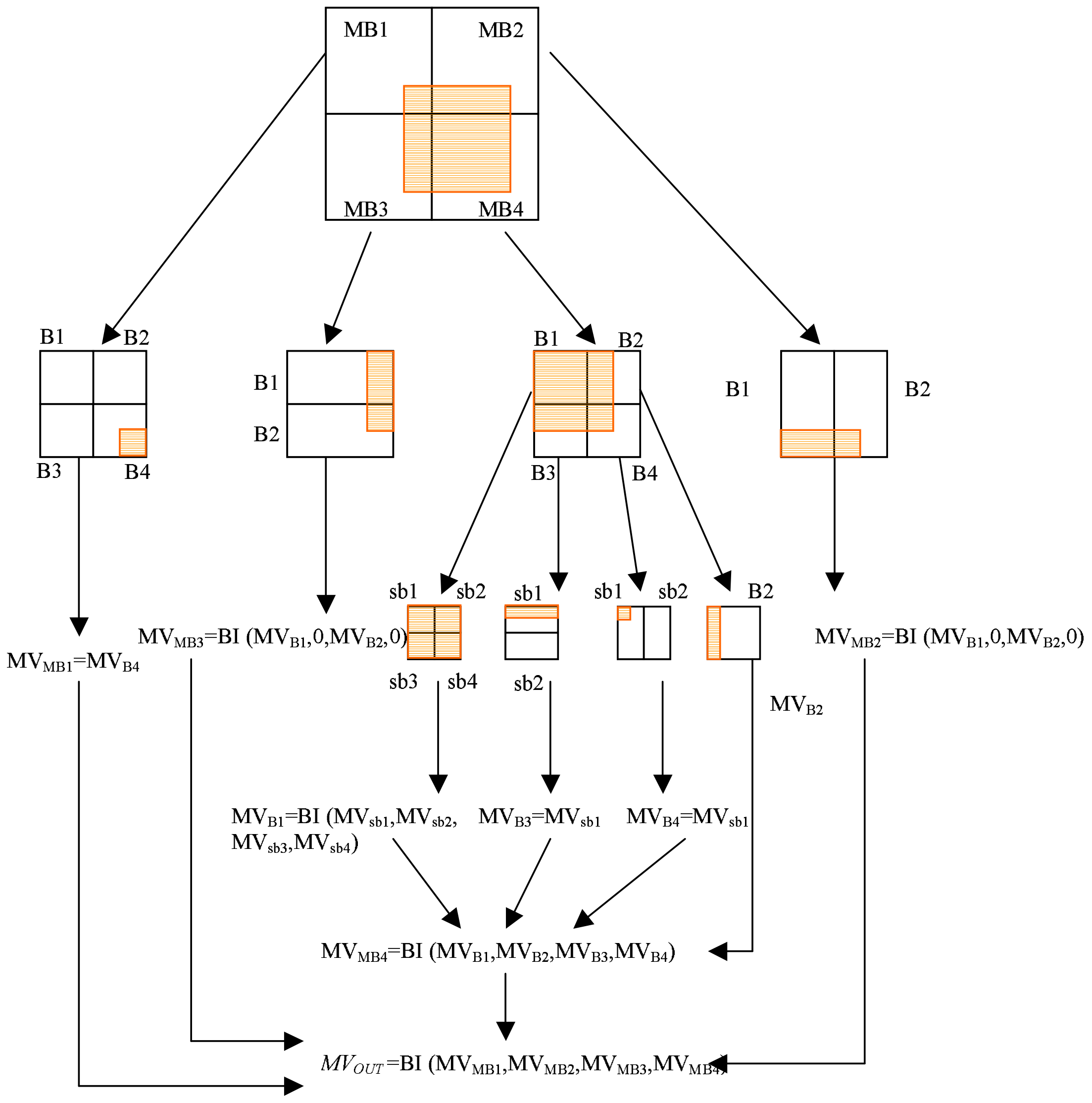

MB1
MB2
MB3
MB4
B1
B2
B3
B4
B1
B2
B1
B2
B3
B4
B1
B2
sb1
sb2
sb3
sb4
sb1
sb2
sb1
sb2
B2
$MV_{MB1}=MV_{B4}$
MV_{MB3}=BI (MV_{B1},0,MV_{B2},0)
MV_{MB2}=BI (MV_{B1},0,MV_{B2},0)
MV_{B2}
MV_{B1}=BI (MV_{sb1},MV_{sb2}, MV_{sb3},MV_{sb4})
$MV_{B3}=MV_{sb1}$
$MV_{B4}=MV_{sb1}$
MV_{MB4}=BI (MV_{B1},MV_{B2},MV_{B3},MV_{B4})
MV_{OUT}=BI (MV_{MB1},MV_{MB2},MV_{MB3},MV_{MB4})

Figure 5.9: Multi-Level Bilinear Scheme (MLBS)

- The considered macroblock has *one partition overlapping the reference area.* The motion vector of this macroblock (16 × 16 pixels) is given back;

- The considered macroblock has *two partitions overlapping the reference area* (each one of 16 × 8 or 8 × 16 pixels). The motion vectors of these two partitions are composed according to Equation 4.8. The obtained motion vector is given back;

- The considered macroblock has *four partitions or blocks overlapping the reference area* (each one of 8 × 8 pixels). For each of these blocks, its partitions overlapping the reference area are considered at the third level, in the following way:

  - The considered block has *one partition overlapping the reference area.* The motion vector of this block (8 × 8 pixels) is given back;
  - The considered block has *two partitions overlapping the reference area* (each one of 4 × 8 or 8 × 4 pixels). The motion vectors of these two partitions are composed according to Equation 4.8. The obtained motion vector is given back;
  - The considered block has *four partitions or sub-blocks overlapping the reference area* (each one of 4 × 4 pixels). The motion vectors of these four partitions are composed according to Equation 4.8. The obtained motion vector is given back.

For each macroblock overlapping the reference area in the skipped frame, our scheme goes down to lower levels, until all partitions and sub-partitions overlapping the reference area are considered. At each level, and for each partition overlapping the reference area, it chooses a motion vector. This is the motion vector of the partition overlapping the reference area, or a composition of the motion vectors of sub-partitions overlapping the reference area. The composition is about at most four motion vectors and is performed according to Bilinear Interpolation Function (4.8). At each level, the composed motion vectors are given back. At the end, four motion vectors related to the four macroblocks overlapping the reference area, are composed according to Equation 4.8. The result of this composition (MV_{OUT} in Figure 5.9) is added to the current motion vector, to obtain the new motion vector pointing to the last not skipped frame. Note that, by using this scheme, it is possible to apply other functions instead of the Bilinear Interpolation. We show in the next section the performance of the proposed scheme. At the best of our knowledge, not exists a motion vector composition algorithm for H.264 temporal transcoding, so the proposed algorithm is a simple technique that can be used to perform motion vector computation by considering variable partition of H.264 frames with a low time complexity. The time complexity is very important in a real-time setting.

5.5.3 Multi-Level Bilinear Scheme performance

We evaluate the PSNR1 and the time complexity of the proposed approach. PSNR1 measure is computed as explained in Section 5.4.

The results presented in this section are related to a temporal transcoder that we implemented by using the H.264/MPEG-4 AVC Baseline Profile [ISO05] [ITU05], with the following setting of parameters: UVLC, variable block-based ME/MC, quarter-pixel MC, one reference frame, motion vector search range of Motion Estimation equal to 16 pixels. The temporal transcoder architecture that we used for implementing this temporal transcoder is described in Section 5.2. The first frame is compressed as an INTRA frame, and all the other ones as INTER frames. The target application is a real-time communication, and so to keeping the minimum latency, bidirectional frames and RDO mode decision are not considered. We assume that, in our transcoding, the variable partitition of frames is the one imposed by front encoder and the motion vectors are re-computed with the MLBS scheme proposed in Section 5.5.2, when the reference frame is skipped. After Motion Vector Composition, a motion vector refinement process of ± 2 pixels around the composed motion vector is applied. To run our experiments, we used an Athlon 64 3200+, with 512MB memory and Linux Mandrake 10.2 as operating system.

We show here the experimental results about "foreman", "akiyo", "coastguard" and "carphone" video sequences with 300 frames, QCIF format, 30 fps. In Table 5.6 we show PSNR1 for different transcoder input bit rates when the same frames are skipped, then the transcoder output bit rate is not constant. PSNR1 of our algorithm is compared with that of Full Search Motion Estimation (FSME), and the Fast Motion Estimation algorithm (MEFast) implemented in the H.264 reference software. We can observe that for all video sequences the proposed method achieves a good performance with respect to the other metrics. In Table 5.7 the complexity of the proposed method, compared with that of FSME and MEFast is shown. The values indicate, for the different input bit rates, the ratio of computation time of the entire transcoding process with our algorithm and MEFast relative to FSME. They are the average values obtained by running the same experiment 20 times. We can observe that our method outperforms in all cases MEFast and it has a great gain with respect to FSME. For "foreman" video sequence we show the performance of the proposed algorithm for low and high frame rate reductions: from 30 to 15 fps (Figure 5.10) and from 30 to 7.5 fps (Figure 5.11), respectively. In both frame rate reductions, the proposed method achieves a good performance with respect to the other metrics. Mainly for high frame rate reductions (30 to 7.5 fps in Figure 5.11), the PSNR1 of our algorithm is close to that of FSME and MEFast.

In Figure 5.12, the complexity of the proposed method for "foreman" video sequence and both frame rate reductions is showed. The reported values are obtained by averaging the computation times of all bit rate reductions cases. We can observe that our method outperforms MEFast and it has a gain of about 40% with respect to FSME. Compared with performance results of Bilinear Interpolation algorithm presented in Section 5.4, the proposed scheme achieves a better PSNR1 with a greater time complexity. This is also due to different used codecs.

Table 5.6: MLBS algorithm performance: PSNR1 of different video sequences at different input bit rate IR(Kbps) from 30 to 15 fps

Algorithm	IR=32	IR=64	IR=128	IR=256	IR=512
foreman					
FSME	33.02	34.39	35.91	37.73	40.28
MEFast	32.96	34.15	35.71	37.61	40.14
MLBS	31.69	33.26	34.97	37.05	39.69
akiyo					
FSME	44.96	46.90	49.97	51.77	51.51
MEFast	44.54	46.80	49.85	51.70	51.46
MLBS	44.41	46.33	49.18	51.61	51.16
coastguard					
FSME	34.31	34.74	35.47	36.54	38.50
MEFast	34.22	34.71	35.40	36.55	38.43
MLBS	33.72	34.20	35.22	36.52	38.44
carphone					
FSME	35.47	36.20	37.46	39.26	40.03
MEFast	35.41	36.17	37.45	39.16	39.99
MLBS	34.83	35.85	36.98	38.88	39.78

Table 5.7: MLBS algorithm performance: Complexity ratio of different video sequences at different input bit rate IR(Kbps) from 30 to 15 fps

Algorithm	IR=32	IR=64	IR=128	IR=256	IR=512
foreman					
FSME	1	1	1	1	1
MEFast	0.81	0.80	0.79	0.85	0.87
MLBS	0.47	0.67	0.65	0.56	0.61
akiyo					
FSME	1	1	1	1	1
MEFast	0.76	0.73	0.76	0.80	0.77
MLBS	0.42	0.63	0.45	0.48	0.48
coastguard					
FSME	1	1	1	1	1
MEFast	0.81	0.82	0.84	0.78	0.85
MLBS	0.42	0.44	0.48	0.52	0.56
carphone					
FSME	1	1	1	1	1
MEFast	0.80	0.83	0.82	0.83	0.81
MLBS	0.43	0.64	0.48	0.52	0.59

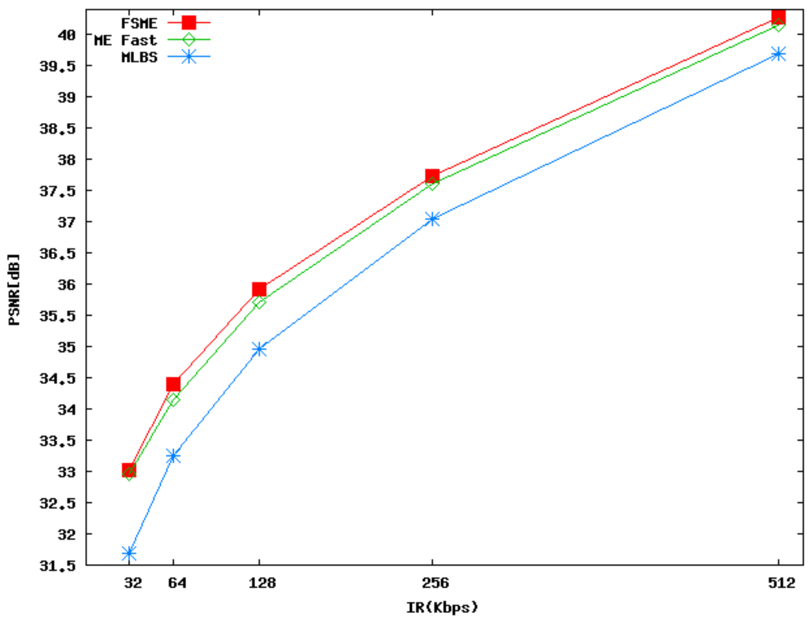


Figure 5.10: MLBS performance: PSNR1 of "foreman" video sequence (from 30 to 15 fps)

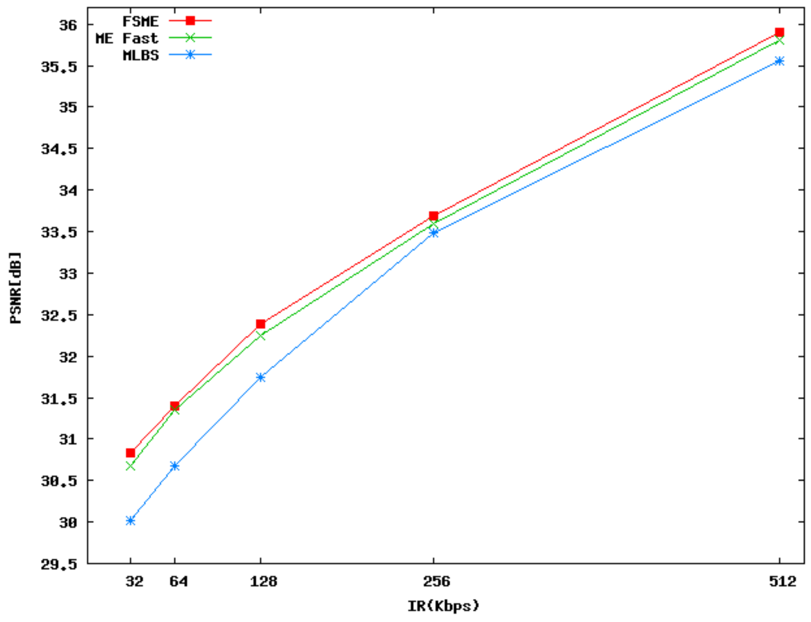


Figure 5.11: MLBS performance: PSNR1 of "foreman" video sequence (from 30 to 7.5 fps)

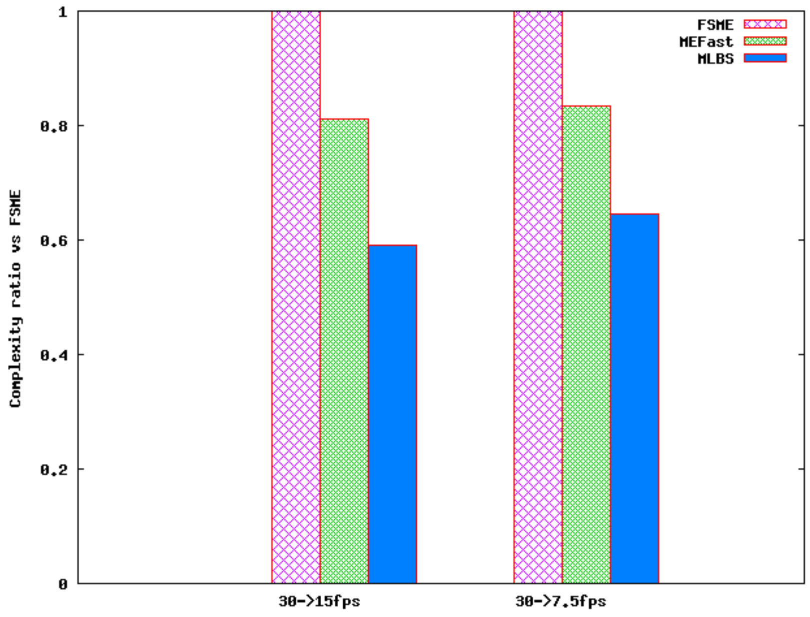


Figure 5.12: MLBS performance: complexity ratio of "foreman" video sequence

Chapter 6

Temporal Transcoding: skipping policies

Abstract

In this chapter, we address temporal transcoding to improve real time video communication in infrastructured networks. We present our results about frame skipping policies. We define the assumed communication model and we formulate the buffer based frame skipping problem. We propose other skipping policies to improve video quality of transcoded sequence when buffer constraints are met. Finally, we propose a skipping policy able to reduce the processing delay of skipping process.

6.1 Introduction

In mobile infra-structured systems, transcoding is needed at interworking nodes to deal with the heterogeneity of the communication infrastructure itself, and with the diversity of services and user terminals. The goal of transcoding is to operate on coded streams whenever there is a set of new constraints different from those assumed when the signals were originally coded. Since many multimedia services are not specifically tailored to mobile systems, often the channel bandwidth required for transmission does not match mobile applications. Besides, the link bandwidth of wireless communications networks is narrow and varies significantly over time. For instance, different bandwidth can be dynamically assigned to a user when its mobile terminal moves to different parts of the cell, or changes its base station during the handoff process. Then, in heterogeneous network systems, the bandwidth of a coded video stream must be drastically reduced in order to cope with a highly constrained transmission channel. Since a great deal of bit rate reduction can be required, traditional methods based on re-quantization are not able to allow a constant frame rate without producing unacceptable distortion in the reconstructed signals. This distortion causes very low quality of service (QoS) delivered to the end user.

Temporal transcoding, by skipping frames, allows having an acceptable quality of transcoded frames, when a high bit rate reduction is needed also. The most important challenge in temporal transcoding is to have a good video quality in real-time communications, such as video telephony or video conferencing. To guarantee real-time constraint of many multimedia services, the transcoding process must provide a minimum processing delay and a communication delay compliant with the time requirements of such services.

In this chapter, we propose some solutions for temporal transcoding dealing with the needs of real time video communication in infra-structured networks, when a constant bit rate reduction is in order.

The buffer control is a well know approach, often used in bit rate reduction algorithms, to face the constant bit rate bandwidth requirements of network applications [AG98]. However, encoding and quality transcoding rate controls use buffer fullness for finding bit allocation and quantization step sizes of frames of video sequences and for applying frame skipping control [SWA03] [SA04]. Notice that, at best of our knowledge, with the only exception of [CSA03], where a frame skipping control scheme based on a buffer level prediction algorithm is presented, other works [HWL98] [FCS02] [SC04] [DKD06] approach the frame skipping problem in temporal transcoding by considering others metrics, mainly motion activity, to improve the quality of transcoded video sequence. However, because of new challenges introduced in video communications (i.e., real-time constraints, high bit rate reduction in mobile systems), buffer control is the most important factor when we want to have a good quality of transcoded video sequences in real-time services. All these reasons motivate us to investigate buffer constraints in frame skipping. The main goal of the proposed skipping policies is to improve the quality of transcoded video sequences in real-time applications. This is mainly obtained by guaranteeing a transmission delay of transcoded frames compliant with the time requirements of real-time systems. In addition, a frame skipping policy, also able to reduce the processing delay of transcoding system, is proposed.

After defining the communication model assumptions common to all the proposed policies, we present a basic skipping policy, based on transcoder output buffer occupancy, that we call"Buffer-occupancy" skipping policy. We describe this policy in detail in Section 6.3.2. In addition, we propose other policies able to improve the video quality of the transmitted video sequence, when transcoder buffer constraints are met. They consider other metrics, such as a new motion activity measure, the number of consecutive skipped frames, and a random choice. Those skipping policies are presented in Section 6.2.2, 6.2.3, and 6.2.4, respectively. Finally, in Section 6.2.5, a faster skipping policy is proposed to greatly reduce the computation time of transcoding process. This skipping policy is also based on transcoder output buffer occupancy.

6.2 Communication model assumptions

The most important part in our temporal transcoding architecture is the Frame Rate Control (FRC) module, depicted in Figure 5.2. In FRC module, frame skipping policies are applied to decide on skipping or transmitting the current frame.

Before describing the proposed skipping policies, we define the assumed communication model.

We assume a communication model where the video is captured and coded with a constant frame rate at the front encoder, the end decoder decodes and displays video frames continuously at the same rate that have been captured at the sender. A system end-to-end delay, that we call D, is assumed. It is the interval between the time a video image is captured by the video camera and encoded at the sender side, and the time it is decoded and displayed on the monitor at the receiver. A frame received after such maximum admitted delay D is not displayed, and it is replaced at the decoder by the previous one, for having a continuous playing. This end-to-end delay D encompasses a processing delay and a network communication delay. The former includes the computation time spent for capturing and compressing at the encoder, for decompressing and displaying at the decoder, and finally for performing transcoding, if needed. The latter is the time needed to move data in the network, including specific protocol delays.

This delay is strongly dependent on the adopted network system, and required service quality. In 3GPP, the definition of a real-time conversational service using speech states that the end-to-end delay should be less than 400 ms [3GP06]. Real-time video is rather different in terms of characteristics compared to real-time voice. The variation in delivery timing of video does not produce the same dramatic effect as for voice, where high delay can trigger concealment actions. The variation in video frame rate produces minor degradation in the video conversational quality because a delayed frame can be replaced with the previous one with an satisfactory quality. Then, the end-to-end delay for video delivery can be higher than 400 ms, maintaining however the synchronization with speech [CFPS07].

Seen in that respect, it is clear that the transcoding delay cannot be too large, considering also that a minimum part of the total end-to end delay is always consumed by the transport mechanism.

In this chapter we investigate the transcoding delay. Then, we distinguish in *transcoding* delay, between a *computation delay*, needed to re-compute motion vectors and residual of a frame when its reference frame is skipped, and a *transmission delay* of transcoded frames, that is the time that a frame remains in the transcoder output buffer waiting for transmission.

In Section 6.2.5, we describe a frame slipping policy able to reduce the *computation delay* in transcoding, needed to re-construct a frame when the previous one is skipped.

In this chapter, our main goal is to investigate frame skipping policies able to guarantee a maximum admitted *transmission delay*, in networks systems with a constant output bit rate (CBR), when a reduction of input bit rate is needed.

We will show in the next chapter, a temporal transcoding application for variable bit rates (VBR) networks.

As we specified in Section 5.2, in our transcoder architecture, the words "input" and "output" are always related to the transcoder. IR and R are the input and output bit rate respectively; ρ indicates the frame rate of the input video sequence. The word "transcoded" is related to a frame that is re-constructed, according to the previous not skipped frame, and put in the transcoder output buffer to be transmitted.

We investigate buffer constraints involved in frame skipping process for guaranteeing a maximum *transmission delay* of transcoded frames.

We define S and L as the size and the occupancy of the transcoder output buffer, respectively. With $l(f)$, we denote the size of the transcoded frame f.

We assume that the buffer occupancy decreases at a constant rate of R/ρ bits every $1/\rho$ seconds. Note that, as we explained in Section 5.2, our temporal transcoder reduces the constant input bit rate IR only by skipping frames, so that the output bit rate R turns out to be constant. Due to skipping of frames, the transcoder output frame rate is not constant, and we assume that the skipped frames are replaced by the previous ones (freezing) at the displaying time in the end decoder.

Disregarding variable transmission delay due to wireless channel errors, the *transmission delay* of the transcoded frames, introduced in the communication between transcoder and end decoder, is determined by L/R. In this way, the maximum *transmission delay*, that we call τ, incurred by a data bit of the transcoded video sequence, is S/R. We can have different values of τ by setting different values of S. This value of S can be set according to the system end-to-end delay needed for guaranteeing real-time communication. In Section 6.3.1, we show the performance obtained by setting different values of τ. Note that τ is the maximum *transmission delay* introduced for a transcoded frame, when the transcoder buffer is full. In the average case, this *transmission delay* is lower than τ.

Two buffer thresholds, B_{lower} and B_{upper}, are established for avoiding buffer underflow and overflow. Underflow occurs when the transcoder output buffer occupancy is zero, and so the end decoder receives data of a frame after it is scheduled to be displayed, causing the stop of the video sequence (besides the waste of the communication bandwidth). Buffer overflow occurs when the buffer occupancy exceeds the buffer size, and it increases the 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.

In Section 6.3, we show the performance of frame skipping policies for different values of the parameters of this communication model, in particular for several values of IR and R.

To have an idea of this communication model, look at Figure 6.1.

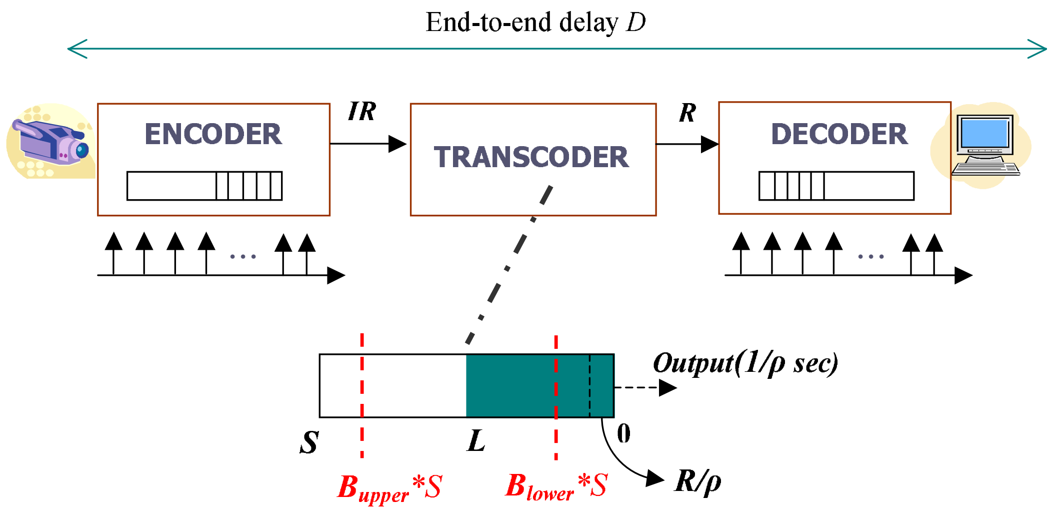


Figure 6.1: Skipping policies: communication model

6.2.1 Buffer-occupancy skipping policy

We present here a buffer-based frame skipping policy that skips frames in order to meet the buffer constraints defined in the above communication model [BLM05a]. In the remainder of this thesis, we call this policy buffer-occupancy. In this policy, 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 of the value of the threshold $B_{upper}S$, in our buffer-occupancy policy, we avoid the buffer overflow by testing that the size of the transcoded frame does not exceed the free buffer space. The first frame, that is an INTRA frame, is always transcoded. If the size of the first frame exceeds the buffer size, we have an additional delay called 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 frame transmission delay τ for the whole transmission.

The whole procedure of buffer-occupancy skipping policy is described by a pseudo-code in Table 6.1.

In Section 6.3, we show the performance of this skipping policy in terms of number of transcoded frames and PSNR of a transcoded video sequence.

When buffer constraints are met, other skipping metrics can be addressed. We propose other skipping policies that take into account buffer constraints similarly to buffer occupancy policy and, in addition, they consider other metrics for improving the quality of the transcoded video sequence. We present such policies in the next sections.

Table 6.1: Pseudo-code of Buffer-occupancy skipping policy

```
**Buffer-occupancy Policy**(frame $f$):
1. if ($f$ = first frame) transcode $f$
2. else
3.   if (($L \leq B_{lower}(S)$)&($L + l(f) \leq S$)) transcode $f$
4.   else
5.     if ($L \geq B_{upper}(S)$) skip $f$
6.     else
7.       if($L + l(f) \geq S$) skip $f$
8.       else transcode $f$
```

6.2.2 A New motion based skipping policy

The drawback of temporal transcoding is the jerky effect caused by skipping of frames, which is more evident when frames with a lot of motion are skipped. The motion activity of a frame is an important issue addressed in literature in the choice of frames to be skipped [HWL98][FCS02][SC04]. In Section 4.5.1, we overview some well known motion activity measures. At the best of our knowledge, no work on skipping policies focus on the type of motion in a frame. That means, no work considers the video quality degradation introduced in the transcoded video sequence by skipping a frame with a great number of small motion vectors, or by skipping a frame with a small number of large motion vectors.

We observe that in known policies [HWL98][FCS02], where the motion activity is computed as the sum of the motion vectors of a frame, the frames with many small motion vectors have a greater motion activity value than those ones with one large motion vector. For example, if in a frame, the motion vectors of all macroblocks (99 macroblocks in QCIF format) are small (their value is equal to 0.5 with half pixel resolution), the sum of these motion vectors is greater than the value of the maximum possible motion vector (its value being 16 in MPEG4 codec, and it corresponds to the search range of Motion Estimation procedure). This implies that, in the existing motion activity based policies, a frame with only one object in movement is probably skipped, even if this movement is fast. For example, in the well known "table tennis" benchmark video sequence, the frames where only the tennis ball moves, are skipped in the motion based skipping policies mentioned so far. But if many consecutive frames with few objects in movement are skipped, and the movement is fast, the jerky effect is more evident. For example, in "table tennis" video sequence, if all frames in which only the tennis ball moves are skipped, the ball disappears in the transcoded video sequence, so producing at the displaying time a visual quality degradation to final user, even if the PSNR measure is not that much decreased. We propose a skipping policy, which avoids such video quality degradation, by introducing a new motion activity measure able to consider the fast movement of few small objects in a frame also. This result has been published in [LM06].

The basic idea of this new measure is that we want to assign the same importance in the video sequence to frames with few great motion vectors and to frames with many small motion vectors. The proposed motion activity measure is the following one:

$$MA_f = \frac{1}{N_{MB}} \sum_m k^{|x_m|} + k^{|y_m|} \tag{6.1}$$

where m is a macroblock, k is a properly tuned constant, x_m and y_m are the motion vector components of macroblock m, and N_{MB} is the number of macroblocks in frame f. In order to find the proper value of the constant k, we impose the following:

$$numMB(k^{\min MV} + k^{\min MV}) \approx (k^{\max MV} + k^{\max MV}) \tag{6.2}$$

where $numMB$ is the number of macroblocks in a frame, $minMV$ and $maxMV$ are the minimum and maximum size of motion vectors components in the macroblock, respectively. We search the value of k for which the motion activity of a frame with only one motion vector having components with the maximum possible value is nearly equal to that of a frame where all components of motion vectors have the minimum possible value. In this way, we obtain a motion activity measure that assigns to frames with few but large motion vectors the same weight of those ones with many small motion vectors. In Section 6.3, we present the performance achieved by tuning the constant k according to Equation 6.2.

We also take into account INTRA macroblocks in the motion activity computation, by assigning 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. The reason for this is that an INTRA macroblock is produced when there are many residuals, and the macroblock is largely different from the reference area in the previous frame.

The proposed motion activity measure is compared with a dynamic threshold Thr in our frame skipping policy. Such threshold, differently to that proposed in [HWL98] and [FCS02], is set as follows:

$$Thr(f) = \frac{\alpha MA(f-1) + (1-\alpha)\sum_{i=2}^{F} MA(f-i)}{NumFrames} \tag{6.3}$$

$$NumFrames = \frac{NumCodedBits}{AverageFrameSize} \tag{6.4}$$

$$AverageFrameSize = \frac{IR}{\rho} \tag{6.5}$$

where α is a properly tuned constant, F is the number of frames transcoded in the current second, $MA(f-1)$ is the motion activity of the previous transcoded frame, $NumFrames$ is the number of frames that would be transcoded in the temporal interval (1 second) if all frames had the same size. $NumFrames$ is defined by Equation 6.4, where $NumCodedBits$ is the total number of coded bits in the temporal interval, and $AverageFrameSize$ is defined by Equation 6.5, where IR is the input bit rate and ρ is the frame rate. Note that, in our motion activity measure, only the motion activity of transcoded frames is considered because the transcoded frames are the only displayed frames at receiver side. Experimentally, we observed that a good value for the constant α, that implies a better visual quality of the transcoded frames, is 0.6. In this way, we want to assign in Equation 6.3, a weight greater than the one of motion activities of all single earlier frames, to the motion activity of the previous frame. This helps in reducing the jerky effect. In our skipping policy, a frame f is transcoded if its motion activity (defined by Equation 6.1) is greater than the threshold $Thr(f)$.

We show in Table 6.2 the pseudo-code of this policy. We can observe that, before a frame is transcoded, buffer thresholds B_{lower} and B_{upper} are tested as in Table 6.1. Only when buffer thresholds B_{lower} and B_{upper} are met, the motion activity measure defined in 6.1 is considered. In this way, this frame skipping policy is able to deal with real-time constraints, while decreasing the jerky effect caused by skipping of frames. In Section 6.3, we will show that this skipping policy outperforms the motion activity policy proposed in [HWL98].

6.2.3 Consecutive skipping policy

We developed this policy for attempting to overcome a harmful problem arising in *hard transcoding conditions*, that is when a high variation between the input and the output bit rate occurs (from 128 kbps to 32 kbps, for instance)[BLM05a]. 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 one increases, since its motion vectors are obtained by adding those ones of the skipped frames, and its residual is high, since it is computed with respect to the last reference frame, that is likely more different from the current one, if many previous consecutive frames are skipped. 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.

Table 6.2: Pseudo-code of New motion based skipping policy

```
New Motion Activity Policy(frame $f$):
 1.  if ($f$ = first frame)
 2.    transcode $f$
 3.    update $Thr(f)$
 4.  else
 5.    if (($L \leq B_{lower}(S)$)&($L + l(f) \leq S$))
 6.      transcode $f$
 7.      update $Thr(f)$
 4.    else
 5.      if ($L \geq B_{upper}(S)$) skip $f$
 6.      else
 7.        if($L + l(f) \geq S$) skip $f$
 8.        else
 9.          if ($MA(f) < Thr(f)$) skip $f$
 10.           else
 11.             transcode $f$
 12.             update $Thr(f)$
```

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 to this problem, by trying to minimize the number of consecutive skipped frames. This is done by forcing the transcoder to drop a frame (even if its transcoding does not cause buffer overflow), in order to prevent that many frames are dropped later.

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

This strategy forces the transcoder to skip $\Gamma - 1$ consecutive frames, in order to prevent the number of consecutive skipped frames to become larger than $\Gamma - 1$.

We show in Table 6.3 the pseudo-code of the whole strategy. As in Table 6.1, buffer thresholds B_{lower} and B_{upper} are considered before skipping or transcoding a

Table 6.3: Pseudo-code of Consecutive skipping policy

```
Consecutive Policy(frame $f$):
1.  if ($f$ = first frame)
2   transcode $f$
3.  numConsecutiveSkippedFrames=0
4.  else
5.    if (($L \leq B_{lower}(S)$)&($L + l(f) \leq S$))
6.      transcode $f$
7.      numConsecutiveSkippedFrames=0
8.    else
9.      if ($L \geq B_{upper}(S)$)
10.       skip $f$
11.       numConsecutiveSkippedFrames ++
12.     else
13.       if($L + l(f) \geq S$)
14.         skip $f$
15.         numConsecutiveSkippedFrames ++
16.       else
17.         if (numConsecutiveSkippedFrames $< \Gamma$)
18.           skip $f$
19.           numConsecutiveSkippedFrames++;
20.         else
21.           transcode $f$
22.           numConsecutiveSkippedFrames=0
```

frame.

This policy does not guarantee that the above critical situation never happens, however it is very unlikely. It depends on the video sequence and on input and output bit rates. It is more likely for video sequences with many scene changes and when the input and output bit rates are very high and very low respectively.

6.2.4 Random skipping policy

Randomization is used for studying the behavior of a system when input data do not follow any known law. In our setting, the sizes of incoming frames are variable and it is not possible to assume a certain distribution. This motivated us to try managing the frame skipping in a randomized way. As we saw above, the temporal transcoder

Table 6.4: Pseudo-code of Random skipping policy

```
Random Policy(frame $f$):
 1. if ($f$ = first frame) transcode $f$
 2. else
 3.   if (($L \leq B_{lower}(S)$)&($L + l(f) \leq S$)) transcode $f$
 4.   else
 5.     if ($L \geq B_{upper}(S)$) skip $f$
 6.     else
 7.       if($L + l(f) \geq S$) skip $f$
 8.       else randomNumber = random() % $S$;
 9.         if (randomNumber $\geq L$) transcode $f$
10.         else skip $f$
```

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 [BLM05a]. 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 the pseudo-code of this strategy in Table 6.4. As in the previous skipping policies presented in this chapter, buffer thresholds B_{lower} and B_{upper} are considered before skipping or transcoding a frame.

6.2.5 Size-prediction skipping policy

In temporal transcoding, the size of a transcoded frame increases when many previous frames are skipped. This is so since its motion vectors are obtained by adding those ones of the skipped frames. Besides, its residual is high because it is computed with respect to the last reference frame, that is temporally far from the current one, so is likely more different from it.

We investigated the features of this growth, and we observed experimentally that the size of a frame grows according to a logarithm function given by :

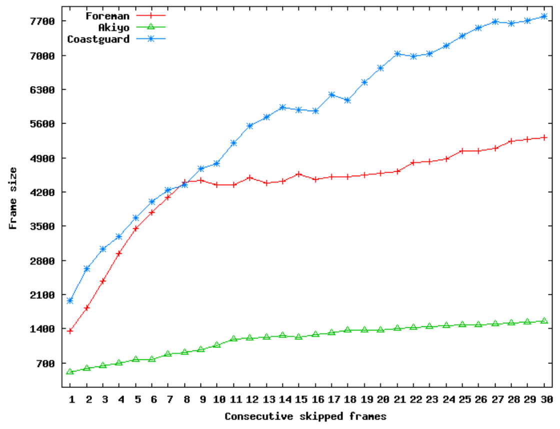


Figure 6.2: Temporal transcoding: size of video frame after skipping a number of previous consecutive frames

$$l(f) = \alpha ln(f + 1) \tag{6.6}$$

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 of those skipped frames. The value of α is dynamically computed each time that the first one of a run of consecutive skipped frames is skipped. For computing α, we use Equation 6.6, where $f = 1$, and $l(f)$ is the computed size of the first skipped frame. This value of α is used in Equation 6.6, to predict the size of a set of consecutive frames, until the last one of those frames is transcoded. Then, when a new frame f will be skipped because its size is larger than the free buffer size, a new value of α will be computed, and will be used in Equation 6.6 for predicting the size of all consecutive frames, until the last of those frames will be transcoded.

In Figure 6.2 we give an idea of this logarithmic growth. The values on the x axis indicate the number n of consecutive skipped frames, and those ones on the y axis, the size of the frame transcoded after n previous consecutive skipped frames.

The reported values in Figure 6.2, refer to “foreman”, “akiyo” and “coastguard” QCIF video sequences coded with MPEG4 with the same quantization parameters (31 and 20 for intra and INTER frames respectively). These results are obtained by starting to skip frames from the $30th$ frame in the video sequences. FDVS algorithm and motion vector refinement of ± 2 pixels used to compute motion vectors.

We propose a Size-prediction policy, applied in the transcoder when a frame of a video sequence is skipped. The basic idea is to avoid the computation needed to reconstruct a frame that will be skipped because its size is higher than the free buffer space. Our approach is to predict the size of the next frames when the current one is skipped. In particular, when the current frame f is skipped, this policy predicts, according to Equation 6.6, the size of the next frame $f + 1$, without computing it.

Table 6.5: Pseudo-code of Size-prediction skipping policy

```
**Size-prediction Policy**(frame $f$):
1.  **if** ($f$ = first frame) transcode $f$
2.  **else**
3.    **if** $((L \leq B_{lower}(S))\&(L + l(f) \leq S))$ transcode $f$
4.    **else**
5.      **if** $(L \geq B_{upper}(S))$ skip $f$
6.      **else**
6.        **if**$(L + l(f) \geq S)$
7.          **do**
8.          skip $f$
9.          predict the size of $f + 1$ according to Equation 6.6
10.         $f = f + 1$
11.         $L = L - (R/\rho)$
12.         **while**$((L > B_{lower}(S))\&(L + l(f) \geq S))$
13.         transcode $f$
```

If this size is higher than the free buffer space, frame $f + 1$ is skipped, and the size of the next frame $f + 2$ is predicted according to Equation 6.6. In this way, when a frame is skipped according to its predicted size, the computation needed for re-constructing motion vectors and residual of such frame is avoided. This result has been published in [BLM05b].

Note that, in our model communication assumptions, buffer occupancy decreases at a constant rate of R/ρ bits every $1/\rho$ seconds.

When the free buffer space is larger than the predicted size of the current frame, this frame is transcoded, and then its motion vectors and residual are computed.

However, as in the skipping policies previously described in this chapter, a frame is transcoded, if the buffer occupancy is lower than a properly tuned threshold, and the predicted size of the frame is lower than the free buffer space, in order to avoid buffer underflow. Note that, each frame is decoded with respect to the last decoded frame. When the policy decides to transcode the frame, if the computed size of this frame is higher than the free buffer size, the frame is skipped.

Compared with the buffer occupancy skipping policy, this one has the advantage of predicting the size of a frame that will be skipped since its size is higher than the free buffer space. Avoiding the computation needed to re-compute frames that will be skipped, this policy greatly reduces the time of the total transcoding process, when many consecutive frames are skipped.

As we will show in Section 6.3, the performance of this policy, in terms of PSNR of the transcoded sequence, is comparable to that of the Buffer-occupancy one, reducing the computation time.

The pseudo-code of our Size-prediction policy is shown in Table 6.5. The equation 6.6 gives a good prediction of the size of a frame, after its previous frames have been skipped. This prediction cannot be appropriate if there is a scene change. In any case, when the predicted size of a frame meets the buffer threshold, the frame is recomputed, and its actual size is checked. If this size is lower than the free buffer space, the frame is transcoded, else it is skipped.

6.3 Performance analysis of skipping policies

In this section, we present a performance comparison of the proposed frame skipping policies. After defining the simulation setting in Section 6.3.1, we evaluate the video quality performance of the proposed skipping policies in Section 6.3.2 and Section 6.3.3.

We present the transcoding processing delay improvement of Size-prediction policy, with respect to Buffer-occupancy skipping policy. Note that, with the exception of Size-prediction policy, the other proposed skipping policies have a computational time comparable to that of Buffer-occupancy one, since the additional operations required by these policies take a minimum time. These operations are needed to compute motion activity in New motion based skipping policy, to generate a random number in Random skipping policy and to compute Γ value in Consecutive skipping policy. The time complexity improvement with respect to Buffer-occupancy policy is significant only for Size-prediction policy because this one avoids transcoding of some frames.

6.3.1 Simulation Setting

The experiments have been performed by considering the transcoding architecture designed in Section 5.2. Forward Dominant Vector Selection (FDVS) algorithm has been used to perform motion vector composition after skipping of a frame. We observed in Section 5.4, that the performance results obtained with the four motion vectors composition algorithms presented in Section 4.3 and implemented in our architecture are very similar in terms of PSNR and time complexity. Compared with the other implemented motion vector composition algorithms, FDVS achieves a good performance in terms of PSNR with a lower computational complexity. After Motion Vector Composition, a motion vector refinement of ± 2 pixels around the composed motion vector, is applied. The experimental results are about several different benchmark video sequences of 300 frames, input frame rate of 30 fps, QCIF and CIF format. For QCIF format, by using Equation 6.2, we obtained k=1.34, while for CIF format we obtained k=1.47. The experimental results presented in this chapter are obtained by using an MPEG4 based transcoder implementation. To evaluate the performance of the proposed frame skipping policies, we used the following metrics:

- *PSNR* of the transcoded video sequence. The PSNR is computed between the transcoded video sequence and the video sequence decoded after the front encoder (and not with respect to the original sequence). In this way, given that our transcoder is a purely temporal one, we want to estimate the quality degradation due to frame dropping only. In particular, we consider the following two PSNR measures:
  - *PSNR1*: this measure is computed taking into account all frames of the transcoded video sequence, by replacing every frame skipped by the transcoder, with the previous not skipped frame (freezing). We introduced this metric in Section 5.4. This metric allows measuring the actual visual quality perceived by the final user, after transcoding;
  - *PSNR2*: this measure is computed by taking into account transcoded frames only. It indicates the quality of single transcoded frames, without capturing the degradation introduced by frame dropping.
- *Number of Transcoded Frames* in the video sequence, that, for the sake of brevity, is indicated with TF. This value represents the smoothness of the transcoded video sequence. Much larger this value is, much smaller the jerky effect perceived by the final user is.

All the proposed skipping policies take into account buffer fullness to meet real-time constraints, as described above. In our communication model (see Section 6.2), the buffer size is related to the maximum admitted *transmission delay* τ of transcoded frames by the following:

$$\tau = \frac{S}{R} \tag{6.7}$$

where S and R are the buffer size and the output bit rate, respectively. The buffer size, and then the setting of τ, can impact the performance of frame skipping policies. In Table 6.6, we report the number of transcoded frames and the average PSNR1 of Buffer-occupancy skipping policy, for different settings of τ, and different input and output bit rates (IR→R). We observe that Buffer-occupancy skipping policy achieves a good performance, for all bit rate reductions, with a value of τ equal at least to 300 ms. With lower values of τ, its performance decreases at low output bit rates, when a high reduction of input bit rate occurs (i.e 128→32 in Table 6.6). This is why, in this case, frames are coded at high bit rates. So, their size is large while the buffer size is sufficient to accommodate a small number of frames, according to Equation 6.7, and consequently a large number of frames are skipped. This motivates us to set $\tau = 300$ in the remainder of the experiments presented in this thesis.

Table 6.6: Buffer-occupancy skipping policy: TF and PSNR1(dB) of "foreman" video sequence for different IR(Kbps), R(Kbps) and τ values

foreman(QCIF)	τ=500		τ=400		τ=300		τ=250	
IR→R	PSNR1	TF	PSNR1	TF	PSNR1	TF	PSNR1	TF
256→128	31.63	164	31.53	163	31.50	164	31.47	165
128→64	28.61	153	29.88	152	29.83	153	29.94	154
64→32	28.63	136	28.57	137	28.45	137	28.46	138
256→64	26.19	66	26.43	67	25.89	63	25.85	61
128→32	25.03	55	24.99	56	24.46	52	20.34	26

6.3.2 Buffer-occupancy *vs* other skipping policies

For each skipping policy, we report its performance results, for "standard" and "hard" transcoding conditions. With "standard" transcoding conditions we mean a typical situation in which the transcoder output channel has a bandwidth equal to a half of the input bandwidth. With "hard" transcoding conditions we mean a situation in which a high reduction of the bit rate channel occurs. For both cases, we consider high and low bit rates. We show here the experimental results about "akiyo", "table", "carphone", "foreman" and "coastguard" video sequences with 300 frames, QCIF format, 30 fps.

To deal with real-time constraints, buffer occupancy is considered in all proposed frame skipping strategies. Consequently, as it was to be expected, there are not large differences on the PSNR1 and PSNR2 achieved by different frame skipping strategies (see Table 6.7 and Table 6.8).

The number of transcoded frames is equal to the ratio between R and IR, with respect to the initial number of frames. In case of the same ratio between R and IR, the number of transcoded frames decreases with low output bit rates.

In Figure 6.3, we compare PSNR1 of New motion based skipping policy with that of Buffer-occupancy and we observe that the former slightly outperforms the latter. By looking at the Table 6.7 and Table 6.8, we observe that in most cases, New motion based skipping policy outperforms Buffer-occupancy policy in terms of PSNR2 of transcoded video sequence. The reason for this is that the first one considers motion activity in the choice of frames to be skipped, so the selected frames have a better visual quality with respect to the frames selected by Buffer-occupancy policy.

We can also note that, Consecutive skipping policy behaves similarly to Buffer-occupancy policy, in standard and hard transcoding conditions in terms of PSNR2, but by looking at Figure 6.4, we observe that Consecutive policy is better than Buffer-occupancy in term of PSNR1 in hard transcoding conditions. This happens, in our opinion, because the frames are dropped more uniformly. Random and Size-

Table 6.7: Proposed skipping policies: TF, PSNR1(dB) and PSNR2(dB) for different video sequences, IR(Kbps) and R(Kbps) values

τ=300	akiyo(QCIF)			table(QCIF)			carphone(QCIF)		
IR→R	PSNR1	PSNR2	TF	PSNR1	PSNR2	TF	PSNR1	PSNR2	TF
Buffer-occupancy									
256→128	38.15	40.55	159	30.98	35.80	140	32.70	35.80	170
128→64	38.41	40.64	129	29.80	32.86	152	31.40	33.78	172
64→32	36.68	38.00	125	27.99	32.11	144	30.12	32.43	140
256→64	37.93	41.98	84	24.02	32.97	33	29.86	35.01	80
128→32	34.11	39.02	47	22.80	31.87	38	29.20	31.85	76
New Motion based									
256→128	39.13	40.66	159	31.55	36.79	144	33.46	35.90	176
128→64	38.40	40.62	129	30.12	33.64	155	31.93	34.20	176
64→32	36.48	38.03	126	28.58	32.19	148	30.25	32.50	149
256→64	37.98	42.01	86	24.07	33.20	33	30.04	35.66	86
128→32	34.12	39.12	47	22.92	32.40	40	29.23	33.68	78
Consecutive									
256→128	39.16	40.30	156	32.55	36.49	150	32.81	36.92	169
128→64	38.52	41.12	148	31.22	34.23	149	31.13	33.90	170
64→32	35.98	38.00	146	28.23	32.01	150	30.15	31.88	150
256→64	38.26	42.11	60	24.97	33.89	47	30.14	34.45	66
128→32	34.10	39.38	50	22.80	31.80	44	29.43	33.86	60
Random									
256→128	38.13	40.52	158	31.11	35.20	140	32.71	35.20	168
128→64	38.39	40.62	130	29.88	32.90	152	31.40	33.78	172
64→32	36.58	38.03	126	27.90	32.01	142	30.03	32.24	138
256→64	37.90	41.87	82	24.12	32.87	34	29.56	34.80	81
128→32	34.05	38.93	45	22.64	31.67	36	29.03	31.62	72
Size-prediction									
256→128	38.15	40.62	156	31.34	35.30	132	32.72	35.32	169
128→64	38.49	40.73	131	29.80	32.67	150	31.03	33.16	160
64→32	36.67	38.05	126	28.19	32.11	143	30.13	32.51	142
256→64	37.80	41.60	80	24.10	32.80	33	29.76	34.92	85
128→32	34.11	38.65	46	22.60	31.58	35	29.24	31.92	70

Table 6.8: Proposed skipping policies: TF, PSNR1(dB) and PSNR2(dB) for other video sequences, IR(Kbps) and R(Kbps) values

τ=300	foreman(QCIF)			coastguard(QCIF)		
IR→R	PSNR1	PSNR2	TF	PSNR1	PSNR2	TF
Buffer-occupancy						
256→128	31.50	34.86	164	30.63	33.18	164
128→64	29.83	33.06	153	29.69	32.14	159
64→32	28.45	32.13	137	28.55	30.83	132
256→64	25.89	34.30	63	25.75	32.91	58
128→32	24.46	31.96	52	25.14	31.29	49
New Motion based						
256→128	31.47	35.36	161	30.20	33.59	159
128→64	29.90	33.47	147	29.73	32.14	153
64→32	28.76	32.58	130	28.63	30.99	128
256→64	25.98	34.70	62	25.87	32.99	59
128→32	24.71	32.32	52	25.13	31.52	50
Consecutive						
256→128	31.59	34.96	158	30.76	33.04	163
128→64	29.84	33.17	147	29.59	32.00	148
64→32	28.43	32.20	123	28.61	30.90	113
256→64	25.95	34.10	59	26.12	32.91	56
128→32	24.93	31.80	42	25.94	31.16	48
Random						
256→128	30.08	34.80	158	29.25	33.15	155
128→64	28.70	32.63	144	29.30	32.04	150
64→32	28.37	32.23	130	28.15	30.23	114
256→64	25.94	34.23	62	25.75	32.78	58
128→32	24.63	32.11	50	25.14	31.18	50
Size-prediction						
256→128	31.40	34.52	162	30.20	33.10	162
128→64	29.85	33.01	151	29.40	32.01	158
64→32	28.38	32.10	135	28.21	30.50	131
256→64	25.90	34.80	61	25.60	32.15	55
128→32	24.63	31.20	53	25.20	31.30	50

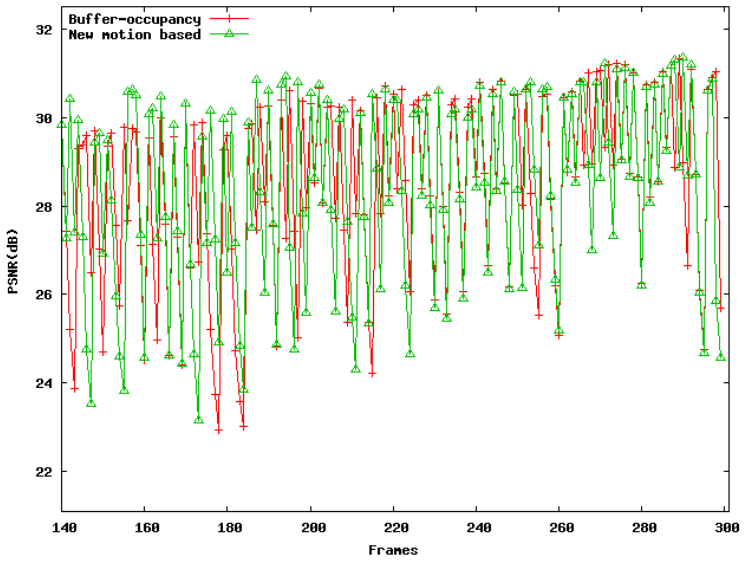


Figure 6.3: Buffer-occupancy *vs* New motion based skipping policies: PSNR1 of "coastguard" video sequence, *IR*=64 Kbps, *R*=32 Kbps

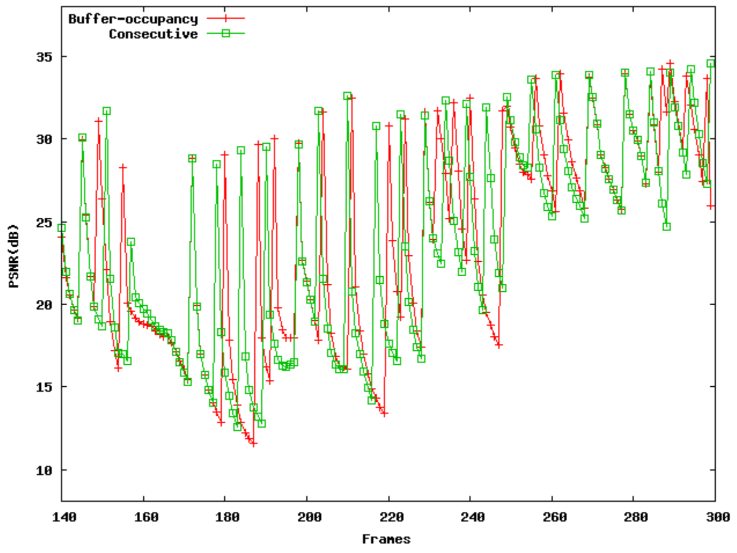


Figure 6.4: Buffer-occupancy *vs* Consecutive skipping policies: PSNR1 of "foreman" video sequence, *IR*=128 Kbps, *R*=32 Kbps

prediction policies behave similarly to Buffer-occupancy policy in terms of number of transcoded frames, PSNR1 and PSNR2 values (see Table 6.7 and Table 6.8). The computation time of Size-prediction policy is much lower (with a decrease of 30-40%), with respect to that of Buffer-occupancy policy, both at low and high bit rates, as depicted in Figure 6.5. The presented results outline that, there is not the best skipping policy and that the most important factor to investigate for real

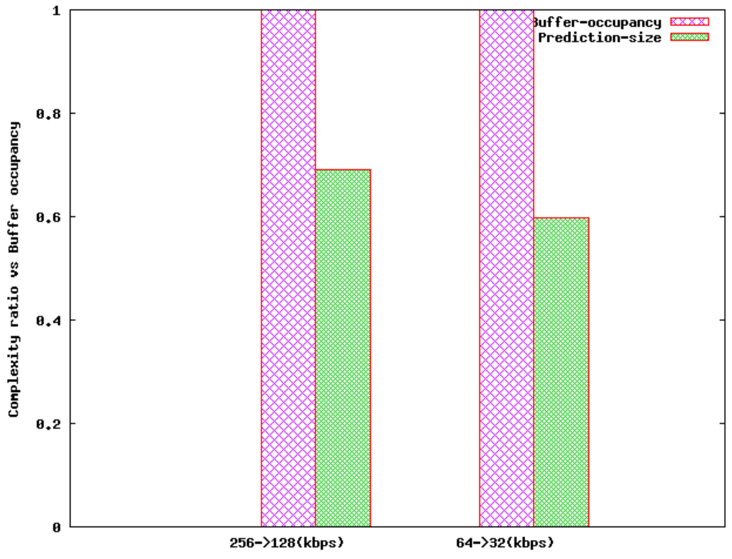


Figure 6.5: Buffer-occupancy *vs* Size-prediction skipping policies: Complexity ratio of "foreman" video sequence

time communication is buffer occupancy over time. This can be done by analytical models also.

6.3.3 New motion based *vs* Standard motion based skipping policy

We evaluated the performance of our New motion based skipping policy (see Section 6.2.2), compared with that of a policy computing motion activity as in [HWL98] [FCS02]. For the sake of clearness, we call the last one "Standard motion based" policy. Note that, we evaluated both skipping policies, with the same threshold defined by Equation 6.3.

By looking at Figure 6.6, we can observe that New motion based policy slightly outperforms Standard motion based policy for video as "mobile", where there are many slowly moving objects. Table 6.9, shows that New motion based policy is better than Standard motion based policy, for several QCIF and CIF video. Its improvement is not so high in terms of PSNR1, PSNR2 and number of transcoded frames. But, by looking at Figure 6.7, we can observe that our motion based policy is able to transcode also frames with few moving objects when this movement is fast (the tennis ball in the second and third pictures of Figure 6.7.(a)); these frames are skipped by Standard motion based policy (Figure 6.7.(b)).

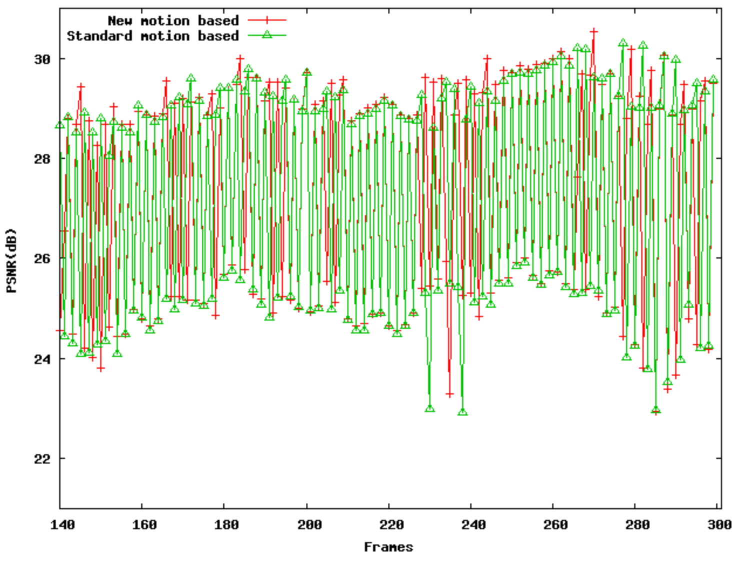
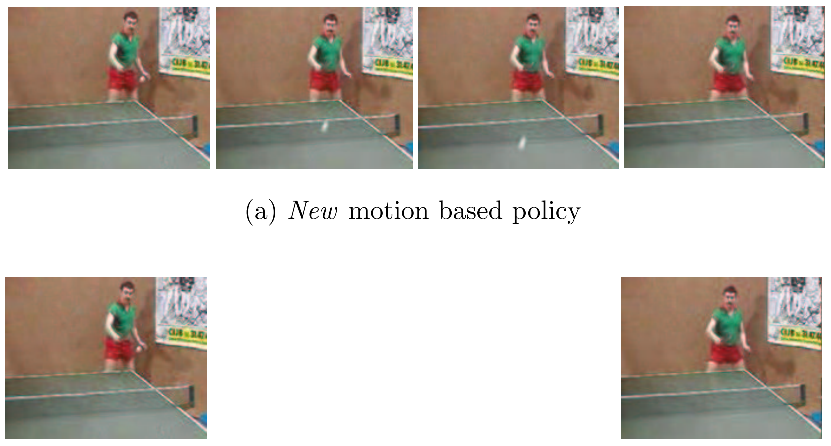


Figure 6.6: New motion based *vs* Standard motion based skipping policies: PSNR1 of "mobile" video sequence, IR=256 Kbps, R=128 Kbps



(a) *New* motion based policy

(b) *Standard* motion based policy

Figure 6.7: New motion based *vs* Standard motion based skipping policies: transcoded frames (210-213) of "table tennis" video sequence, QCIF format, IR=128 Kbps, R=32 Kbps

Table 6.9: New motion based *vs* Standard motion based skipping policies: TF, PSNR1(dB) and PSNR2(dB) for different video sequences, *IR*(Kbps) and *R*(Kbps) values

τ=300	New Motion based			Standard motion based		
IR→R	PSNR1	PSNR2	TF	PSNR1	PSNR2	TF
QCIF Video						
table						
256→128	31.55	36.79	144	31.06	36.30	143
128→64	30.12	33.64	155	29.69	33.12	155
128→32	22.92	32.40	40	22.89	32.22	40
mobile						
256→128	27.67	29.50	164	27.15	29.10	163
128→64	26.85	28.64	162	26.45	28.24	160
128→32	22.93	26.91	60	22.91	26.82	60
carphone						
256→128	33.46	35.90	176	33.02	35.60	174
128→64	31.93	34.20	176	31.60	33.80	175
128→32	29.23	33.68	78	29.19	33.65	78
CIF Video						
foreman						
1024→512	29.30	32.50	170	28.99	32.25	172
512→256	29.20	32.40	163	28.81	31.95	165
512→128	24.05	30.65	64	23.80	29.78	64
mobile						
1024→512	25.86	29.01	180	25.38	28.55	181
512→256	25.56	28.01	176	25.03	27.43	175
512→128	21.63	26.26	81	21.60	26.01	81
coastguard						
1024→512	30.20	33.80	172	29.69	33.30	170
512→256	29.20	31.90	163	28.83	31.81	161
512→128	24.19	31.24	58	24.01	31.02	57

Chapter 7

Temporal Transcoding in IEEE 802.11 Vehicular Networks

Abstract

In this chapter, we address the real time video transmission issues in ad hoc networks, based on IEEE 802.11 protocol. We show that temporal transcoding is a good solution to deal with bandwidth reductions and channel access delay, in vehicular networks.

7.1 Introduction

In the last years, the growth of wireless local area networks based on the IEEE 802.11 standard represents a practical network solution offering mobility, flexibility, low cost and easy deployment, as well as providing a mobile access to Internet. This encourages the use of this protocol for new nomadic applications. Among such applications, wireless inter-vehicle networks could improve road security and offer new driving services [YLZV04]. This large diffusion brings out the need to realize advanced mobile multimedia services for a variety of professional and personal uses of such networks.

The IEEE 802.11 MAC protocol defines a set of rules for mobile stations to gain access to shared wireless medium through two different access mechanisms: DCF (Distributed Coordination Function) and PCF (Point Coordination Function). PCF was originally developed for supporting real-time traffic, and QoS enhancements of this scheme was developed [AKC04], but it is rarely adopted by current commercial products because of its implementation complexity. The DCF mode is widely used. IEEE 802.11 DCF is a contention based channel access protocol. A shared wireless medium is randomly accessed by contentions among stations in a service area. Accordingly, the access delay increases significantly with the number of contending stations.

On the other hand, real-time video communication has inherent delay-sensitive characteristics. Each video packet must arrive before its pre-defined deadline. If it arrives after this deadline, it is seen as a late packet, and it is discarded at the receiver. However, under DCF, a station might have to wait an arbitrarily long time to send a frame, if there is network congestion. So, real time applications such as voice and video may be severely degraded.

Many existing works focus on MAC layer solutions for having high quality video transmission in IEEE 802.11 DCF networks. In [KT06][ZWC03][LSC05], new channel access protocols, based on timestamp of video packets and retransmission of them, are proposed. Moreover, enhancements of DCF are developed in the new IEEE Draft, the so called IEEE 802.11e [IEE04], defining the MAC protocol to support LAN applications with QoS requirements, including the transport of voice, audio and video over WLANs. In this thesis, we are interested in the basic version of IEEE 802.11 DCF function as presented in Section 7.2, because it has a minor complexity and it is the most used. Other works address the problem of delay and video quality degradation due to channel errors over IEEE 802.11 based networks, proposing solutions to reduce the number of discarded video packets, and their subsequent re-trasmissions [MBM05]. In our system assumptions, the channel errors are not considered, since we concentrate on the MAC protocol effects on the available bandwidth.

Cross-layer optimization has been recently proposed for improving the performance of real-time video transmission over 802.11 WLANs. In [HTL$^+$05], a cross layer signaling between the MAC layer and the video encoder is proposed. There, a MAC adaptation method adjusts the MAC/radio parameters according to packet loss statistics, so that optimal quality of packet transmission is achieved. Then, a video encoder, interacting with the MAC layer, changes quantization parameters in order to adapt the video rate to link quality. A cross-layer solution has been also proposed to perform video transcoding over IEEE 802.11 with access points in [CMP$^+$05]. There, an adaptive algorithm, considering both instantaneous network throughput and the video sequence characteristics, is proposed to perform video streaming. If the bit-rate must be reduced, the algorithm selects whether to perform frame size reduction and/or frame rate reduction, with a thresholds based mechanism.

The original contribution of the present thesis to this research area is to propose a temporal transcoding system improving real-time video communication in IEEE 802.11 ad hoc networks in terms of bandwidth saving when congestion occurs and video quality displayed at receiver side. To the best of our knowledge, no work investigates the advantage of using the temporal transcoding approach to reduce bandwidth needs when congestion occurs, consequently avoiding delayed packet delivery and video quality degradation. The proposed method takes into account the behaviour of the basic DCF function to enhance video quality delivery. We approach the problem by considering IEEE 802.11 vehicular networks. This type of networks is characterized by the strong mobility of the vehicles, very dynamic topology and a short communication time. Our solution is able to guarantee real time constraints of video delivery in vehicular networks, where real time communication is required for enhancement of road safety by propagating emergency alerts or entertainment applications. This contribution has been published in [BGLM07].

However, our approach can be easily extended to consider other IEEE 802.11 based networks, instead of vehicular networks.

After describing the DCF function in Section 7.2, we address real-time video transmission in IEEE 802.11 based vehicular networks in Section 7.3, and describe our approach in Section 7.4. Finally, in Section 7.4.3, the performance evaluation of our system is shown compared with that of the conventional IEEE 802.11 scheme.

7.2 IEEE 802.11 Distributed Coordination Function

In recent years, much interest has been involved in the design of wireless networks for local area communication. IEEE has standardized the 802.11 protocol for Wireless Local Area Networks. In the IEEE 802.11 protocol, the primary medium access control (MAC) protocol is called Distributed Coordination Function (DCF). This is a carrier sense multiple access with collision avoidance (CSMA/CA) scheme, and binary slotted exponential backoff. The standard also defines an optional Point Coordination Function (PCF), which is a centralized MAC protocol able to support collision free and time bounded services.

We are interested in DCF scheme. In this section, we briefly summarize the features of such scheme, as standardized in [IEE97], which are involved in our communication model assumptions.

DCF describes two techniques to be used for packet transmission. The default scheme is a *two-way handshaking* technique called basic access [Bia00]. This mechanism is characterized by the immediate transmission of a positive acknowledgement (ACK) by the destination station, upon successful reception of a packet transmitted by the sender station. A station with a new packet to transmit monitors the channel activity. If the channel is idle for a period of time equal to a Distibuted Interframe Space (DIFS), the station transmits. Otherwise, if the channel is sensed busy (either immediately or during the DIFS), the station persists to monitor the channel until it is measured idle for a DIFS. At this point, the station generates a random backoff interval before transmitting, to minimize the probability of collision with packets being transmitted by other stations. The time immediately following an idle DIFS is slotted and a station can transmit only at the beginning of each slot time. The slot time size depends on the physical layer and it is set equal to the time needed by any station to detect the transmission of a packet from any other station.

DCF adopts an exponential backoff scheme. At each packet transmission, the backoff time is uniformly chosen in the range $(0, CW)$. The value CW is called con-

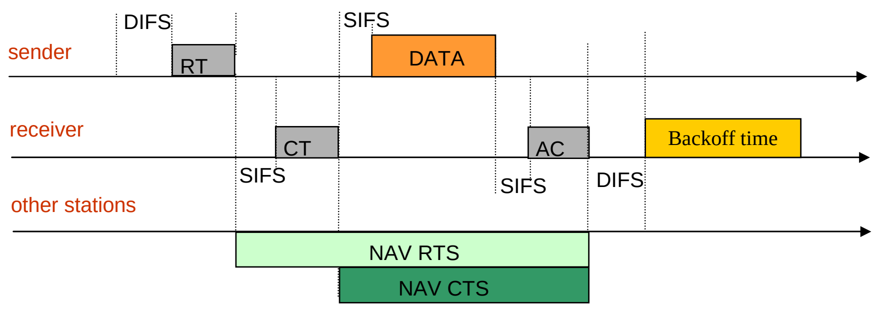


Figure 7.1: IEEE 802.11 Distributed Coordination Function: RTS/CTS Access Mechanism

tention window and depends on the number of transmissions failed for the packet. At the first transmission attempt, CW is set equal to a value CW_{min}, called minimum contention window. The backoff timer is decreased for each time slot the medium remains idle. Otherwise, if the medium is sensed busy, the backoff timer is frozen, and reactivated when the channel is sensed idle again for more than a DIFS. As the backoff timer expires while the medium is idle, the node waits DIFS amount of time before getting access to the channel. In case of collision, communicated through lack of an acknowledgement, the size of the contention window is doubled following Equation 7.1.

$$CW = (CW_{min} \times 2^i) - 1 \tag{7.1}$$

where i is the number of transmission attempts. The value of CW_{min} is physical layer specific. After each successful transmission, the CW is initialized with the CW_{min} value.

Since the CSMA/CA does not rely on the capability of the stations to detect a collision by hearing their own transmission, an ACK is transmitted by the destination station, to signal the successful packet reception. The ACK is immediately transmitted at the end of the packet, after a period of time called Short Interframe Space (SIFS). Since SIFS is shorter than a DIFS, no other station is able to detect the channel idle for a DIFS until the end of the ACK transmission. If the transmitting station does not receive the ACK within a specified ACK timeout, or it detects the transmission of a different packet on the channel, it reschedules the packet transmission according to the above backoff rules.

In addition, DCF defines an additional *four-way handshaking* technique to be optionally used for a packet transmission [Bia00]. This mechanism, known as RTS/CTS access mechanism, is shown in Figure 7.1.

A station that wants to transmit a packet, waits until the channel is sensed idle for a DIFS. Then, follows the above backoff rules, and instead of the packet, at first transmits a special short frame called Request To Send (RTS). When the receiving station detects an RTS frame, and it is available for receiving a packet, it answers, after a SIFS, with a Clear To Send (CTS) frame. The transmitting station is allowed to transmit its packet only if the CTS frame is correctly received. In RTS and CTS frames there is the information about the length of the packet to be transmitted. This information can be read by any listening station, which updates a Network Allocation Vector (NAV) containing the information of the period of time in which the channel will be busy. The RTS/CTS mechanism reduces the length of the frames involved in the contention process. In fact, assuming a perfect channel sensing by every station, collision may occur only when two or more packets are transmitted within the same slot time. If both transmitting stations use the RTS/CTS mechanism, collision occurs only on the RTS frames and they can early detect such collision by the lack of CTS response.

7.3 Real-time video on IEEE 802.11 vehicular networks

Inter-vehicular wireless networks are gaining much attention in the research community due to the number of applications that could improve the quality of everyday life. On one hand, such applications include vehicle-to-vehicle (V2V) communications, for delivering of safety information (state of the road after an accident, or fog, or snow, for instance). General Motors recently announced the development of the V2V technology (called "sixth sense") able to warn drivers about critical situations. Data, such as location and speed, are exchanged inter-vehicles, and instantly processed to promptly inform the driver of possible danger [Mar06]. In Europe, the Car-2-Car Communication Consortium, jointed both by car and electronics manufacturers and academic institutions, is working towards standardization of an inter-vehicle communication technology based on IEEE 802.11 wireless LAN. The goal is to create and establish an open European industry standard for Car2Car communication systems, and to guarantee European-wide inter-vehicle operability, by promoting the allocation of a royalty free European wide exclusive frequency band for Car2Car applications. The aim is to increase road traffic safety and also efficiency. They foresee the release of a full specification standard for the end of 2008 [car]. On the other hand, there are communication, information and also entertainment applications, such television, audio/video telephony, gaming, etc., possibly involving access to Internet, that improve the comfort and productivity of passengers inside the vehicles by offering an ubiquitous access. Experts in the field, and economists, believe that a high number of vehicles will be equipped with V2V technology before 2015 [Mar06].

Protocols specifically aimed at inter-vehicular communications have been proposed in [Cse98][OYAC99]. These solutions, however, require the development of new standards and devices. In our approach, we adopt currently available wireless networking protocols, such as the widely used IEEE 802.11 Wireless Local Area Network standard, as presented in the previous section, without any changes. This makes a large and not expensive application of our system suitable. IEEE 802.11 technology for Wireless Local Area Networks (WLAN) seems to offer a suitable way for deploying inter-vehicular communication networks [Mar06] [car][OK04][BMK+05]. Vehicular networks based on IEEE 802.11 with access points have been proposed in the past in [OK04], where TCP and UDP traffics in vehicles moving at different speeds and passing one or more access points at the roadside have been evaluated.

In this thesis, we are interested in ad hoc networks, that allow low-cost communication for mobile users. The development of an infrastuctured wireless network for vehicles could be very expensive and requires very long deployment times. Vehicular ad hoc networks (VANET), instead, require low cost equipment to be mounted on cars.

The transmission of real time video sequences in vehicular ad hoc networks, incur in some problems, in particular due to transmission errors, the variable bandwidth, and the channel access delay, that bring to packet loss and variable delay of delivering.

Video transmission in vehicular ad hoc network has been studied in [GAZ05] [BMK+05]. In [GAZ05], an architecture to support video streaming applications is proposed. That architecture is composed of two parts: a video source trigger subsystem, and a video data transfer sub-system. The first is responsible for forwarding video trigger messages to the destination region, so that vehicles activate the video camera for capturing video. The second sub-system sends video data back to the node which requested an image from the destination region: this is done by means of a store-carry-and-forward scheme.

In [BMK+05], video transmission experiments between two cars equipped with IEEE 802.11 devices for two typical driving scenarios are presented, by varying bitrates and packetization policies. Then, an algorithm which adapts the video packet size to the current driving scenario (in highway scenario, large packets lead to better performance, while in the urban scenario, small packet sizes are better) is proposed.

7.4 Temporal transcoding on IEEE 802.11 vehicular networks

The main purpose of vehicular networks research is to enable applications that guarantee road safety by delivering data and video emergency warnings about traffic and road scenarios [YLZV04]. There are real-time video services that require a great bandwidth, having strict delay requirements. The goal is to achieve low latency in delivering emergency video warnings, and efficient bandwidth usage.

We propose a solution improving real-time video transmission in vehicular networks, by reducing bandwidth consumption. In the following, we describe the assumptions and the behavior of our system.

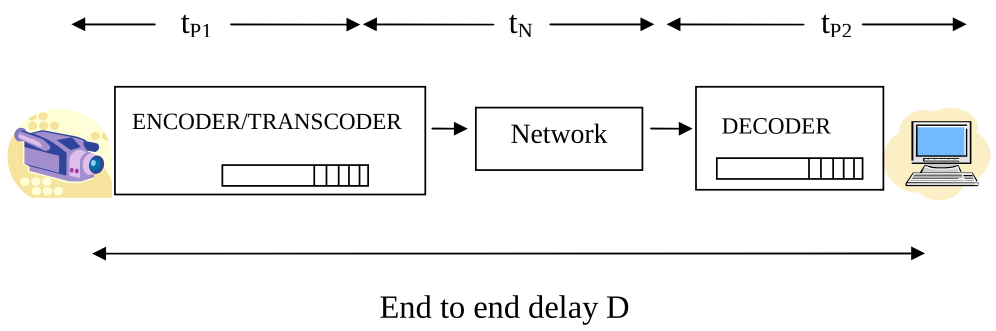


Figure 7.2: IEEE 802.11 wireless network: communication model

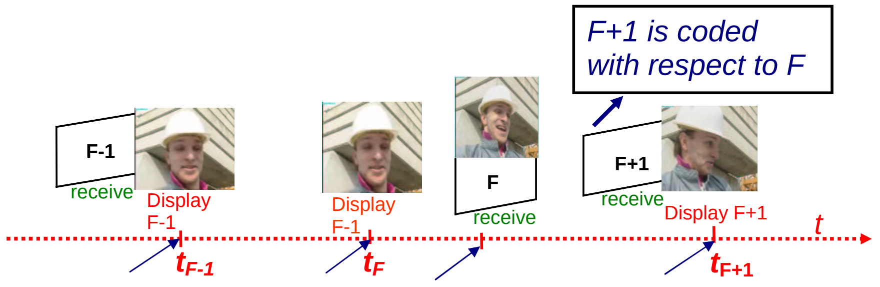
7.4.1 System assumptions

We guarantee real-time communication by setting a maximum admitted end-to-end delay D between sender and receiver nodes. We design a communication model where the video is captured and coded with a constant frame rate at the sender node, and the receiver displays video frames continuously at the same rate that have been captured at the sender. The system end-to-end delay is the interval between the time a video image is captured by the video camera at the sender side, and when it is displayed on the monitor at the receiver. We depict this model in Figure 7.2. A frame received after the maximum admitted delay D is not displayed, and it is replaced by the previous one, for having a continuous playing. Note that, this happens when one or more packets, containing data of a frame, are late also, and not only when the whole frame is late.

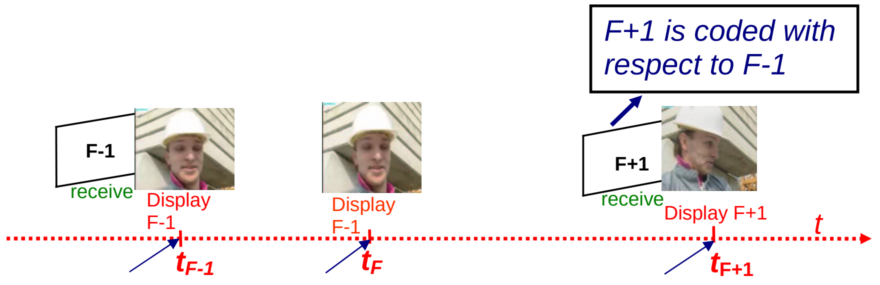
The delay D encompasses a processing delay (denoted with t_P) and a network delay (called t_N). The former includes the time spent for capturing, coding, and transcoding if it is needed, at the sender (t_{P1} in Figure 7.2), and for decoding and displaying at the receiver (t_{P2} in Figure 7.2). The latter is the time needed to move data in the network, including protocol delays. Assuming that the processing delays t_{P1} and t_{P2} for a frame are known and they assume values depending on the adopted coding/transcoding algorithms and the used computation power, fluctuations in the value of D are due to the network delay t_N only. In Section 7.4.3, we show that this network delay is high, due to network congestion and delay in access channel with medium network traffic also, and it implies a greatly decreased quality of video sequence at the receiver.

Note that, the communication model depicted in Figure 7.2 is similar to that one presented in Section 6.2, the main difference being in this model that no output buffer is needed in the transcoder to perform frame skipping in order to have a constant bit rate, since a variable bit rate is supported by the network below.

According to the IEEE 802.11 DCF function presented in Section 7.2, we can distinguish a transmission delay (that we call t_T) and a channel access delay (that we



(a) without temporal transcoding



(b) with temporal transcoding

Figure 7.3: IEEE 802.11 wireless network: decoding process at the receiver side

call t_C) in the network delay (t_N). The former is due to DIFS and SIFS times added to packet transmission and RTS/CTS ones, if the *four-way handshaking* mechanism is used. The latter is the time that a node needs to get the shared channel resource. While a node can know the transmission delay t_T for each packet, the channel access delay t_C greatly depends on the network traffic. When network congestion occurs, t_C can be very high, and greatly increases the end-to-end delay of video packets. In the next section, we explain the basic idea of our system that allows to overcome this problem.

7.4.2 Our approach

As explained in Section 7.2, in DCF *four-way handshaking* mechanism, only when the medium is sensed idle for a DIFS time, the station can start its transmission. At this point the station knows that the time needed to transmit the packet is equal to the transmission delay (that we call t_T). All the same, the station may need to defer its transmission several times because another station begins the transmission during the DIFS or the backoff period. When network load is high or even moderate, the defer time may be several seconds or even higher [ZWC03]. In IEEE 802.11, there is no timeout for deferring. For real-time multimedia applications, this will cause performance degradation, since any frame received after the deadline is not displayed. So, in DCF, it is possible to transmit late frames when network load is moderate or heavy. A late frame is a frame received after the maximum admitted end-to-end delay. This delay is due to time needed to gain access to the channel in order to transmit the frame. Transmission of late frames has two effects: one is wasting bandwidth, the other is delaying the successive frames, which further degrades the performance of real-time applications.

The goal of our approach is to guarantee that a late frame, is not transmitted. The main idea of the proposed system is that each sender node monitors the channel access delay t_C for a video frame transmission. When this delay becomes so high (this is due to deferred transmission since the medium is busy) that the frame will exceed the maximum end-to-end admitted delay D, the frame is skipped, and the next frame will be transcoded with respect to the last transmitted one. If skipping occurs on two consecutive frames, the sender assumes congested network and decides to reduce the frame rate. By simulations, we observed that when two or more consecutive frames are late, there is network congestion and it is needed to skip many consecutive frames before solving congestion, with a high degradation of visual quality. By reducing frame rate it is possible to overcome congestion with a better video quality. Our system is able to detect network congestion at MAC level, and to apply a sort of congestion control at application level, by using temporal transcoding. Some kinds of cross layer techniques can be assumed to allow interaction between different network levels [SYZ+05][BDM+04][CMP+05]. In our system, applying temporal transcoding avoids transmission of late frames. This has two advantages: bandwidth saving with consequent congestion reduction, and enhancement of video quality by avoiding many consecutive late frames. Note that, in the assumed communication model described above, late frames (received after the maximum admitted end-to-end delay) cannot be displayed at the receiver node, and they are used only for decoding next frames.

In Figure 7.3, it is shown what happens at the receiver side, by applying temporal transcoding or not. If transcoding is not applied (Figure 7.3.(a)), frame F is received after the time it is scheduled for displaying, and so it is used only for decoding frame $F+1$. With transcoding (Figure 7.3.(b)), frame F is not transmitted while transcoded frame $F+1$ is. Then, the receiver decodes frame $F+1$ with respect to $F-1$. Thanks to temporal transcoding, the quality of frame $F+1$ in this last case, is nearly as good as that one of frame $F+1$ when transcoding is not applied. So, the advantage of applying transcoding is to save the bandwidth needed to transmit F.

In Section 7.4.3, we will show that our system achieves a good performance in terms of video quality of received frames and bandwidth consumption, with respect to IEEE 802.11 standard protocol based networks. This is because our approach avoids that many consecutive frames are received after the maximum end-to-end admitted delay, and they are not displayed.

Table 7.1: Pseudo-code of temporal transcoding system on IEEE 802.11

```
$\mathbf{D} = t_P + t_N$
$t_N = t_T + t_C$
$t_T = \mathbf{DIFS} + t_{RTS} + \mathbf{2SIFS} + t_{CTS} + t_{packet}$

1. if ($t_C \geq D - t_P - t_T$) for frame $F$
1.     skip frame $F$;
2.     transcode frame $F$+$1$;
3.     if ($t_C \geq D - t_P - t_T$) for frame $F - 1$
4.         halve frame rate.
```

In Table 7.1, we resume the basic idea of the proposed system. As showed in this table, we reduce frame rate by halving it because halving frame rate is more simple than reducing it of an arbitrary measure. For the same reason, halving frame rate is used in encoding process also. (For example, it is less complex converting 30 fps into 15 fps than converting 30 fps into 20 fps).

Note that, our system allows overcoming the congestion problem, when it occurs at the sender node. In a multihop system, only the sender node can perform transcoding, and the forward of packets to destination station is relied on routing mechanisms. So, the sender node is not able to detect congestion that occurs at one intermediate node between the sender and the receiver. We plan to improve our approach in case of real time bidirectional communication, by considering a well know *piggybacking* technique [KR05]. In this way, the sender node receiving a video frame transmitted from the destination node can detect that the one's own is received before the maximum admitted delay at the destination node. If, in a bidirectional communication, the sender node, after sending a frame, does not receive a frame by the destination before a timeout, it assumes that congestion occurs in an intermediate node in the route. If this happens for two consecutive frames, as in the proposed solution, the sender node halves the frame rate. For unidirectional communication, a delay larger than D, can be tailored.

We present experimental results of the proposed approach, without this possible improvement, in Section 7.4.3.

7.4.3 System performance

Due to the novelty of vehicular network applications, simulation is the most effective tool available to the research community for evaluating protocols and architectures.

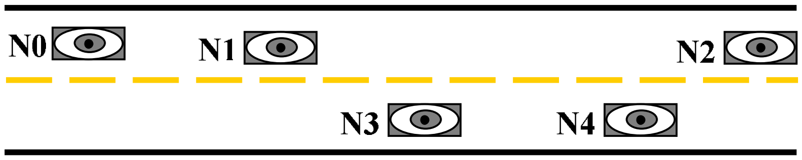


Figure 7.4: Vehicular network: simulated road scenario

Providing realistic road scenarios is important in order to have results close to real world deployment. We evaluated different traffic situations, and we present the results about a typical road scenario, shown in Figure 7.4. Two experiments are reported with this road scenario: the first one, considers only UDP based traffic, including video and voice, while the second one considers both UDP and TCP based traffic.

In the following, we shall describe the setting adopted for our simulations, and then we shall report the results of the proposed approach.

7.4.3.1 Simulation settings

The proposed approach was implemented using NS2 network simulator [ns2]. The channel physical characteristics follow the specification of IEEE 802.11b, with channel bit rate of 11 Mbps. The radio transmission range was set to 250 meters, and all nodes have a directional antenna. The underlying MAC protocol is based on IEEE 802.11 DCF, with RTS/CTS handshaking mechanism. The values of t_{RTS} and t_{CTS} are 352 and 304 microseconds respectively. DIFS and SIFS are equal to 200 and 100 microseconds respectively, and Ad hoc On-Demand Distance Vector (AODV) protocol [PBRD03] is used for packets routing. In our experiments, nodes shown in Figure 7.4, move with different speeds: N0, N1, N2 have a speed of 120 Km/h; N3, N4 have a speed of 90 Km/h; nodes start transmission approximately at the same time, setting a random start time within an interval of 0.3 sec.

The reported values are obtained by setting the maximum admitted delay, D, equal to 100 ms, according to [Bal00]. Our arguments are still valid with lower values of D.

We used the well known benchmarks “foreman”, “coastguard” and “akiyo” CIF video sequences (10 sec), coded with MPEG4 standard codec at 30 frames/sec (fps), with an intra period of 300 frames.

We evaluated the performance of the proposed system in terms of PSNR of a video sequence between the original video sequence and the sequence displayed at the receiver. We computed the average PSNR by considering all frames, where each missing frame is replaced by the previous one. In addition, we report the best and worst PSNR values occurred in one second of the video sequence, estimated by considering a sliding window of 30 frames (1 sec). We set quantization parameters (QP) for intra and INTER frames, as reported in Table 7.2, in order to have a

Table 7.2: QP and PSNR of different video sequences after coding and decoding

video flow	Intra/Inter QP	Avg(dB)	Best(dB)	Worst(dB)
foreman	20/6	35.72	37.71	33.88
coastguard	20/7	33.71	34.00	33.29
akiyo	20/4	41.18	41.32	40.93

good compromise between video quality of coded sequences, and their bandwidth usage. In the same table, for comparison purposes, we report the PSNR values of the above video sequences after coding and decoding, without considering the effects of network transmission.

We performed the following two experiments with the road scenario depicted in Figure 7.4:

- *first experiment*: we have one flow of voice traffic of 10 Kbps between N0 and N1, and five flows of video traffic, transmitted at initial frame rate of 30 fps, with maximum packet size of 2300 bytes ("foreman" between N0 and N2 and N2 and N0, "coastguard" between N3 and N4 and N4 and N3, "akiyo" between N1 and N2);

- *second experiment*: we have the same voice and video traffic described above for the first experiment, plus a continuous TCP based traffic between N0 and N1, with packet size equal to 1000 bytes.

7.4.3.2 Simulation results

For both experiments, we report the PSNR results obtained by considering the conventional IEEE 802.11 standard network (Table 7.3 and Table 7.4). As we can see, in both cases, there is a high number of frames that exceeds the maximum admitted delay (Expired in Table 7.3 and Table 7.4), due to network congestion, if our approach is not applied. We observed that most of these frames are consecutive, and this is the cause of video quality degradation. A quantization of the number of consecutive skipped frames is showed in Figures 7.5 and 7.6, where, if conventional IEEE 802.11 is used, PSNR values of many consecutive video frames, that are late and so not displayed, are very low.

In our system, we avoid that many consecutive frames exceed the maximum admitted delay. This is done by halving the frame rate when two consecutive frames are late. Transcoding of single late frames improves the quality of video sequences, as we can see in Figures 7.5 and 7.6. More specifically, the use of temporal transcoding allows PSNR values very close to that ones of coded/decoded video sequence, until the frame rate reduction is applied. In particular, in the first experiment, it is needed to halve the frame rate of two video flows and for these video flows, we note a bandwidth required reduction (N0→N2 and N2→N0 in Table 7.5). We note that, for the video flow N0→N2 (Figure 7.5), if transcoding of single late frame is applied, the frame rate reduction occurs at approximately the 240th frame. This implies a higher PSNR value of the whole video sequence.

From the second experiment, it is obvious that the TCP traffic implies higher channel contention. So, in our system, frame rate reduction is more frequent. In fact, it is needed to halve the frame rate of four video flows and for these video flows we note a bandwidth required reduction (N0→N2, N2→N0, N3→N4 and N1→N2 in Table 7.6). In particular, starting from initial frame rate of 30 fps, we obtain a frame rate of 7.5 fps for three video sequences (N0→N2, N2→N0, N1→N2) and 15 fps for the fourth one (N3→N4), if transcoding is applied (Table 7.6).

Nevertheless, even with TCP traffic, our approach outperforms IEEE 802.11 standard protocol based networks, in terms of PSNR values. In addition, our approach allows also a bandwidth saving of the video flows with reduced frame rate.

These experimental results show that our system achieves a better video quality, and a bandwidth saving, compared to IEEE 802.11 standard protocol based networks. We plan to evaluate the performance of the proposed system by considering a dynamic adjustment of frame rate according to network traffic conditions, including an increase of frame rate when the network traffic lowers.

Table 7.3: Conventional IEEE 802.11 protocol: PSNR *vs* required bandwidth of video flows with network congestion (I experiment)

video flow	Avg(dB)	Best(dB)	Worst(dB)	Expired	BW(Kbit)
N0 → N2	23.42	36.00	11.34	175	7904
N2 → N0	18.56	33.94	11.42	230	7904
N3 → N4	19.88	34.00	13.84	224	10255
N4 → N3	33.61	34.00	33.11	3	10255
N1 → N2	39.66	41.32	28.78	41	2401

Table 7.4: Conventional IEEE 802.11 protocol: PSNR *vs* required bandwidth of video flows with network congestion (II experiment)

video flow	Avg(dB)	Best(dB)	Worst(dB)	Expired	BW(Kbit)
N0 $\longrightarrow$ N2	19.13	33.98	11.48	222	7904
N2 $\longrightarrow$ N0	18.59	33.95	11.44	232	7904
N3 $\longrightarrow$ N4	19.42	34.00	13.84	237	10255
N4 $\longrightarrow$ N3	33.20	34.00	30.86	13	10255
N1 $\longrightarrow$ N2	33.56	41.26	24.40	159	2401

Table 7.5: Proposed temporal transcoding system on IEEE 802.11 protocol: PSNR *vs* required bandwidth of video flows (I experiment)

video flow	Avg(dB)	Best(dB)	Worst(dB)	BW(Kbit)
N0 $\longrightarrow$ N2	34.82	37.69	29.49	6778
N2 $\longrightarrow$ N0	29.95	32.92	27.27	4721
N3 $\longrightarrow$ N4	33.71	34.00	33.29	10255
N4 $\longrightarrow$ N3	33.56	33.96	32.73	10230
N1 $\longrightarrow$ N2	41.18	41.32	40.93	2401

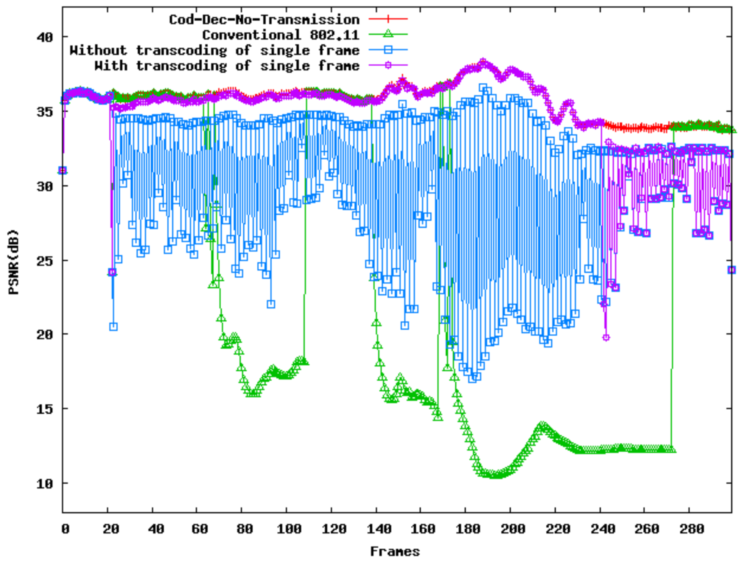


Figure 7.5: Proposed temporal transcoding system on IEEE 802.11 protocol *vs* Conventional IEEE 802.11 protocol: PSNR values of "foreman" video sequence N0$\rightarrow$N2, I experiment.

Table 7.6: Proposed temporal transcoding system on IEEE 802.11 protocol: PSNR *vs* required bandwidth of video flows (II experiment)

video flow	Avg(dB)	Best(dB)	Worst(dB)	BW(Kbit)
N0 → N2	27.64	32.13	23.96	3691
N2 → N0	27.75	31.74	23.96	3678
N3 → N4	29.88	33.78	27.33	7874
N4 → N3	33.71	34.00	33.29	10225
N1 → N2	37.87	41.20	35.97	1471

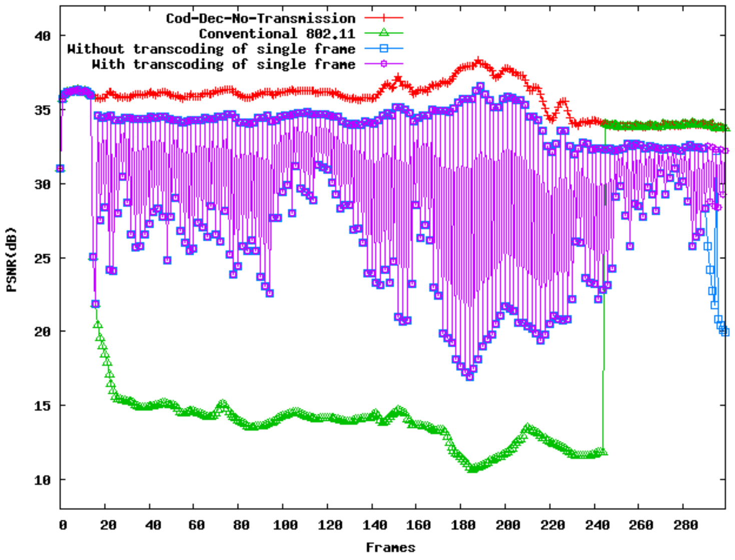


Figure 7.6: Proposed temporal transcoding system on IEEE 802.11 protocol *vs* Conventional IEEE 802.11 protocol: PSNR values of "foreman" video sequence N2→N0, I experiment.

Chapter 8

Conclusions

Abstract

In this chapter, we briefly resume our experience in studying the temporal transcoding problem in mobile systems. We summarize our results, highlighting some open problems and possible directions of future research.

In this thesis, we addressed the temporal transcoding problem in mobile systems. In particular, our goal was to investigate temporal transcoding issues for real-time video communication. We approached the problem by considering firstly an infrastructured network, where transcoding is needed to face bit rate constraints of mobile systems channels. Then, we addressed temporal transcoding to improve real time communications in the emergent vehicular networks setting, where congestion and channel access delay can cause video quality degradation.

In the first part of our work, we concentrated on the design of our temporal transcoder architecture, and its features aimed to improve transcoding efficiency in terms of quality of the video sequence and processing delay of transcoding. So, we addressed Motion Vector Composition, and we proposed a motion vector composition algorithm for H.264 temporal transcoding.

Then, the focus of our work has been to investigate solutions to improve temporal transcoding for real-time video communication in infrastructured networks with constant bit rate reductions. After reviewing the existing solutions in the field, we proposed some frame skipping policies able to meet real-time constraint of our communication model, and to produce a good quality of the transcoded video sequence also when a high bit rate reduction is in order. Temporal transcoding is not the only solution to perform this bit rate reduction. Quality transcoding, by tuning quantization parameters, reduces the bit-allocation of frames to deal with bandwidth reduction. Temporal transcoding provides a video sequence with a lower smoothness due to frame skipping, and a better quality of transcoded frames, if a good frame skipping policy is applied. On the contrary, quality transcoding, avoiding jerkiness, produces a bothering blurry effect in the video sequence, mainly when a high bit rate reduction occurs. In our opinion, the best solution depends on target applications and video sequences features. For sequences with much motion, quality transcoding should be the better solution because without skipping frames it preserves motion of the frames. On the contrary for video sequences with not much motion, temporal transcoding should be the better solution because by skipping some frame it is possible to reduce the output bit rate without decreasing the visual quality if the skipped frames are replaced with the previous ones. Future works include the design of a quality transcoder and a quality-temporal transcoding comparison for facing the problem of real-time video communication in infrastructured networks.

The last part of our work focused on investigating the problem of real time video transmission in vehicular networks. We observed that, the video quality of real time services in these network systems is greatly decreased when there is network congestion, in moderate and heavy traffic situations. We proposed a real-time video transmission system in IEEE 802.11 based vehicular networks applying temporal transcoding at application level to overcome network congestion, detected al MAC level. In particular, the proposed system is able to skip a late frame, that is a frame that does not meet real time constraints due to high access channel delay in its transmission. In addition, our system performs frame rate reduction when two consecutive frames are late. We shown that our system achieves a better video quality and a larger bandwidth saving compared to IEEE 802.11 standard protocol based networks. We plan to evaluate the performance of the proposed system by considering a dynamic adjustment of frame rate according to network traffic conditions, including an increase of frame rate when the network traffic lowers.

Future works in this research area include applying our approach to other ad hoc networks systems such as WLANs, by studying the performance of our system according to mobility models (random way point model, for example), and number of nodes in the network.

Bibliography

[3GP06] 3GPP. *Services and service capabilities. TS 22.105, 3rd Generation Partnership Project(3GPP).* June 2006.

[AG98] P. A. A. Assuncao and M. Ghanbari. A frequency-domain video transcoder for dynamic bit-rate reduction of MPEG-2 bit streams. *IEEE Transactions on Circuits and Systems for Video Technology*, 8(8):953–967, December 1998.

[AKC04] J.N. Al-Karaki and J.M. Chang. A simple distributed access control scheme for supporting QoS in IEEE 802.11 wireless lans. In *Proceedings of IEEE Wireless Communications and Networking Conference (WCNC)*, volume 1, pages 213–218, 2004.

[ANR74] N. Ahmed, T. Natarajan, and K.R. Rao. Discrete cosine transfom. *IEEE Transactions on Computers*, C-23(1):90–93, 1974.

[AWSZ05] I. Ahmad, X. Wei, Y. Sun, and Y.Q. Zhang. Video transcoding: an overview of various techniques and research issues. *IEEE Transactions on Multimedia*, 7(5):793–804, 2005.

[Bal00] M. Baldi. End-to-end delay analysis of videoconferencing over packet switched networks. *IEEE/ACM Transactions on Networking (TON)*, 8:479–492, August 2000.

[BC98] N. Bjork and C. Christopoulos. Transcoder architecture for video coding. *IEEE Transactions on Consumer Electronics*, 44(1):88–98, 1998.

[BDM$^+$04] P. Bucciol, G. Davini, E. Masala, E. Filippi, and J. C. De Martin. Cross-layer perceptual ARQ for H.264 video streaming over 802.11 wireless networks. In *Proceedings of IEEE Global Telecommunications Conference*, volume 5, pages 3027–3031, December 2004.

[BGLM07] M. A. Bonuccelli, G. Giunta, F. Lonetti, and F. Martelli. Real-time video transmission in vehicular networks. In *Proceedings of MOVE Infocom Workshop*, May 2007.

[Bia00] G. Bianchi. Performance analysis of the IEEE 802.11 distributed coordination function. *IEEE Journal on Selected Areas in Communications*, 18(3):535–547, March 2000.

[BLM05a] M. A. Bonuccelli, F. Lonetti, and F. Martelli. Temporal transcoding for mobile video communication. In *Proceedings of The Second Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services (MobiQuitous)*, pages 502–506, July 2005.

[BLM05b] M.A. Bonuccelli, F. Lonetti, and F. Martelli. A fast skipping policy for h.263 video transcoder. In *Proceedings of 12th International Workshop on Systems, Signals and Image Processing (IWSSIP)*, pages 353–358, September 2005.

[BMK$^+$05] P. Bucciol, E. Masala, N. Kawaguchi, K. Takeda, and J. C. De Martin. Performance evaluation of H.264 video streaming over inter-vehicular 802.11 ad hoc networks. In *Proceedings of 16th IEEE International Symposium on Personal Indoor and Mobile Radio Communications*, volume 3, pages 1936–1940, September 2005.

[car] Car2car communication consortium. http://www.car-2-car.org.

[CCP02] M.J. Chen, M.C. Chu, and C.W. Pan. Efficient motion-estimation algorithm for reduced frame-rate video transcoder. *IEEE Transactions on Circuits and Systems for Video Technology*, 12(4):269–275, 2002.

[CFPS07] S. Chakraborty, T. Frankkila, J. Peisa, and P. Synnergren. *IMS Multimedia Telephony over Cellular Systems*. Wiley 2007.

[CL99] P.C. Chang and T.T. Lu. A scalable video compression technique based on wavelet transform and mpeg coding. *IEEE Transactions on Consumer Electronics*, 45(3):788–793, August 1999.

[CMP$^+$05] G. Convertino, D. Melpignano, E. Piccinelli, F. Rovati, and F. Sigona. Wireless adaptive video streaming by real-time channel estimation and video transcoding. In *Proceedings of International Conference on Consumer Electronics (ICCE)*, pages 179–180, January 2005.

[CSA02] P.F. Correia, V.M. Silva, and P.A. Assunção. Rate prediction model for video transcoding applications. In *Proceedings of IEEE International Symposium on Telecommunications (1)*, pages 641–644, 2002.

[CSA03] P.F. Correia, V.M. Silva, and P.A. Assunção. A method for improving the quality of mobile video under hard transcoding conditions. In *Proceedings of IEEE International Conference on Communications (26(1))*, pages 928–932, 2003.

[Cse98] C. Cseh. Architecture of the dedicated short-range communications (DSRC) protocol. In *Proceedings of IEEE Vehicular Technology Conference (VTC)(3)*, pages 2095–2099, May 1998.

[DCU$^+$02] S. Dogan, A. Cellatoglu, M. Uyguroglu, A.H. Sadka, and A.M. Kondoz. Error-resilient video transcoding for robust internetwork communications using GPRS. *IEEE Transactions on Circuits and Systems for Video Technology*, 12(6):453–464, 2002.

[DKD06] A.D. Doulamis, D.I. Kosmopoulos, and N.D. Doulamis. Content-based time sampling for efficient video delivery over networks of low and variable bandwidth. In *Proceedings of International Conference on Digital Telecommunications*, page 27, 2006.

[DL96] W. Ding and B. Liu. Rate control of mpeg video coding and recording by rate-quantization modeling. *IEEE Transactions on Circuits and Systems for Video Technology*, 6(1):12–20, 1996.

[FCS02] K.T. Fung, Y.L. Chan, and W.C. Siu. New architecture for dynamic frame skipping transcoder. *IEEE Transactions on Image Processing*, 11(8):886–900, 2002.

[FCS04] K.T. Fung, Y.L. Chan, and W.C. Siu. Low-complexity and high-quality frame-skipping transcoder for continuous presence multipoint video conferencing. *IEEE Transactions on Multimedia*, 6(1):31–46, 2004.

[GAZ05] M. Guo, M.H. Ammar, and E.W. Zegura. V3: A vehicle-to-vehicle live video streaming architecture. In *Proceedings of PerCom*, pages 171–180, March 2005.

[GCK99] M. Gallant, G. Côté, and F. Kossentini. An efficient computation-constrained block-based motion estimation algorithm for low bit rate video coding. *IEEE Transactions on Image Processing*, 8(12):1816–1823, 1999.

[Gha90] M. Ghanbari. The cross-search algorithm for motion estimation (image coding). *IEEE Transactions on Communications*, 38(7):950–953, 1990.

[HTL$^+$05] I. Haratcherev, J. Taal, K. Langendoen, R. Lagendijk, and H. Sips. Fast 802.11 link adaptation for real-time video streaming by cross-layer signalling. In *Proceedings of the International Symposium on Circuits and Systems (ISCAS)(4)*, pages 3523–3526, May 2005.

[Huf52] D.A. Huffman. A method for the construction of minimum redundancy codes. *Proceedings of the IRE*, 40(9):1098–1101, 1952.

[HWL98] J.N. Hwang, T.D. Wu, and C.W. Lin. Dynamic frame-skipping in video transcoding. In *Proceedings of IEEE Second Workshop on Multimedia Signal Processing*, pages 616–621, 1998.

[I.E03] I.E.G.Richardson. *H.264 and MPEG-4 Video Compression, Video Coding for Next-generation Multimedia.* Wiley 2003.

[IEE97] IEEE standard for Wirelss LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications, IEEE Standard 802.11. November 1997.

[IEE04] IEEE 802.11e Wirelss LAN Medium Access Control (MAC) Enhancements for Quality of Service (QoS). 802.11e Draft 8.0. 2004.

[ISO04] ISO/IEC. Information Technology-Coding of audio visual objects-Part 2: Visual International Standard ISO/IEC 14496-2. 2004.

[ISO05] ISO/IEC. Information Technology-Coding of audio visual objects-Part 10: Advanced video coding. International Standard ISO/IEC 14496-10. 2005.

[ITU05] ITU-T. Advanced video coding for generic audiovisual services. Recommendation H.264, International Telecommunication Union. 2005.

[KH04] S. Kim and Y.S. Ho. Rate control algorithm for H.264/AVC video coding standard based on rate-quantization model. In *Proceedings of IEEE International Conference on Multimedia and Expo (ICME)(1)*, pages 165–168, 2004.

[KPKK06] G.R. Kwon, S.H. Park, J.W. Kim, and S.J. Ko. Real-time r-d optimized frame-skipping transcoder for low bit rate video transmission. In *Proceedings of 6nd IEEE International Conference on Computer and Information Technology*, page 177, 2006.

[KR05] J. F. Kurose and K. W. Ross. *Computer Networking: A Top-Down Approach Featuring the Internet.* Addison-Wesley, 3rd edition, 2005.

[KT06] J.-O. Kim and H. Tode. MAC-layer support for real-time video over IEEE 802.11 DCF networks. *IEEE Transactions on Communications*, 89(4):1382–1391, April 2006.

[LCZ97] H.J. Lee, T. Chiang, and Y.Q. Zhang. Scalable rate control for very low bit rate (vlbr) video. In *Proceedings of International Conference on Image Processing*, volume 2, pages 768–771, 1997.

[LH03] W.J. Lee and W.J. Ho. Adaptive frame-skipping for video transcoding. In *Proceedings of International Conference on Image Processing (1)*, pages 165–168, 2003.

[LL01] C.W. Lin and Y.R. Lee. Fast algorithms for DCT-domain video transcoding. In *Proceedings of International Conference on Image Processing (1)*, pages 421–424, 2001.

[LM06] F. Lonetti and F. Martelli. A new motion activity measure in temporal video transcoding. In *Proceedings of 13th International Conference on Systems, Signals and Image Processings and Semantic Multimodal Analysis of Digital Media (IWSSIP06 and COST 292)*, pages 451–455, September 2006.

[LM07a] F. Lonetti and F. Martelli. Motion vector composition algorithm in H.264 transcoding. In *Proceedings of 14th International Conference on Systems, Signals and Image Processing (IWSSIP) and 6th EURASIP Conference Focused on Speech and Image Processing, Multimedia Communications and Services (EC-SIPMCS)*, June 27-30, Maribor, Slovenia, 2007. Accepted for publication.

[LM07b] F. Lonetti and F. Martelli. Temporal video transcoding for multimedia services. *Book chapter to appear in Multimedia Services in Intelligent Environments, G. A. Tsihrintzis and L. Jain Eds, Springer-Verlag*, Germany, 2007.

[LSC05] M-H. Lu, P. Steenkiste, and T. Chen. Video streaming over 802.11 WLAN with content-aware adaptive retry. In *Proceedings of IEEE International Conference on Multimedia and Expo (ICME)*, pages 723–726, 2005.

[LTYB05] X. Lu, A. Michael Tourapis, Peng Yin, and Jill Boyce. Fast mode decision and motion estimation for H.264 with a focus on MPEG-2/H.264 transcoding. In *Proceedings of the International Symposium on Circuits and Systems (ISCAS)(2)*, pages 1246–1249, 2005.

[Mar06] A. Marshall. V2V: GM technology can prevent accidents. http://geneva2007.media.gmeurope.achtg.de/press_gm.php, December 2006.

[MBM05] E. Masala, M. Bottero, and J.C. De Martin. MAC-level partial checksum for H.264 video transmission over 802.11 ad hoc wireless networks. In *Proceedings of Vehicular Technology Conference (VTC Spring)(5)*, pages 2864–2868, 2005.

[MGL05] S. Ma, W. Gao, and Y. Lu. Rate-distortion analysis for h.264/avc video coding and its application to rate control. *IEEE Transactions on Circuits and Systems for Video Technology*, 15(12):1533–1544, December 2005.

[MWS03] D. Marpe, T. Wiegand, and H. Schwarz. Context-based adaptive binary arithmetic coding in the H.264/AVC video compression standard. *IEEE Transactions on Circuits and Systems for Video Technology*, 13(7):620–636, 2003.

[NHK95] Y. Nakajima, H. Hori, and T. Kanoh. Rate conversion of MPEG coded video by re-quantization process. In *Proceedings of International Conference on Image Processing*, pages 408–411, 1995.

[ns2] The network simulator (NS2). http://www.isi.edu/nsnam/ns/.

[OK95] A. Ortega and M. Khansari. Rate control for video coding over variable bit rate channels with applications to wireless transmission. In *Proceedings of International Conference on Image Processing (3)*, pages 388–391, 1995.

[OK04] J. Ott and D. Kutscher. Drive-thru internet: IEEE 802.11b for “automobile” users. In *Proceedings of IEEE INFOCOM (1)*, pages 362–373, March 2004.

[OYAC99] H. Oh, C. Yae, D. Ahn, and H. Cho. 5.8 ghz DSRC packet communication system for ITS services. In *Proceedings of IEEE Vehicular Technology Conference (VTC)(4)*, pages 2223–2227, September 1999.

[PBRD03] C. Perkins, E. Belding-Royer, and S. Das. Ad hoc on-demand distance vector (AODV) routing. Internet Engineering Task Force, RFC 3561, July 2003.

[PK05a] V. Patil and R. Kumar. An arbitrary frame-skipping video transcoder. In *Proceedings of IEEE International Conference on Multimedia and Expo*, pages 1456–1459, 2005.

[PK05b] V. Patil and R. Kumar. A DCT domain frame-skipping transcoder. In *Proceedings of ICIP (1)*, pages 817–820, 2005.

[RK04] I.E.G. Richardson and C.S. Kannangara. Fast subjective video quality measurement with user feedback. *Electronics Letters*, 40(13):799–801, 2004.

[RRCC00] G.D.L. Reyes, A.R. Reibman, S.-F. Chang, and J.C.I. Chuang. Error-resilient transcoding for video over wireless channels. *IEEE Journal on Selected Areas in Communications*, 18(6):1063–1074, 2000.

[SA04] Y. Sun and I. Ahmad. A robust and adaptive rate control algorithm for object-based video coding. *IEEE Transactions on Circuits and Systems for Video Technology*, 14(10):1167–1182, October 2004.

[SALZ06] Y. Sun, I. Ahmad, D. Li, and Y.Q. Zhang. Region-based rate control and bit allocation for wireless video transmission. *IEEE Transactions on Multimedia*, 8(1):1–10, February 2006.

[SC04] H. Shu and L.P. Chau. Frame-skipping transcoding with motion change consideration. In *Proceedings of the International Symposium on Circuits and Systems (ISCAS)*, pages 773–776, 2004.

[SG00] T. Shanableh and M. Ghanbari. Heterogeneous video transcoding to lower spatio-temporal resolutions and different encoding formats. *IEEE Transactions on Multimedia*, 2(2):101–110, 2000.

[SKHK03] K.d. Seo, S.k. Kwon, S. K. Hong, and J.k. Kim. Dynamic bit-rate reduction based on frame-skipping and requantization for MPEG-1 to MPEG-4 transcoder. In *Proceedings of the International Symposium on Circuits and Systems (ISCAS)(2)*, pages 372–375, 2003.

[SLP04] I.h. Shin, Y.L. Lee, and H.W Park. Motion estimation for frame-rate reduction in H.264 transcoding. In *Proceedings of Second IEEE Workshop on Software Technologies for Future Embedded and Ubiquitous Systems*, pages 63–67, May 2004.

[SWA03] Y. Sun, X. Wei, and I. Ahmad. Low delay rate-control in video transcoding. In *Proceedings of the International Symposium on Circuits and Systems (ISCAS)(2)*, pages 660–663, 2003.

[SYZ$^+$05] E. Setton, T. Yoo, X. Zhu, A. Goldsmith, and B. Girod. Cross-layer design of ad hoc networks for real-time video streaming. *IEEE Wireless Communications*, 12(4):59–65, August 2005.

[SZZL01] G. Shen, B. Zeng, Y.Q. Zhang, and M.L. Liou. Transcoder with arbitrarily resizing capability. In *Proceedings of the International Symposium on Circuits and Systems (ISCAS) (5)*, pages 25–28, 2001.

[TG00] K.T. Tan and M. Ghanbari. A multi-metric objective picture-quality measurement model for MPEG video. *IEEE Transactions on Circuits and Systems for Video Technology*, 10(7):1208–1213, 2000.

[VCS03] A. Vetro, C. Christopoulos, and H. Sun. Video transcoding architectures and techniques: an overview. *IEEE Signal Processing Magazine*, 20(2):18–29, March 2003.

[vid] YUV video sequences. Available at http://trace.eas.asu.edu/yuv.

[VYLS02] A. Vetro, P. Yin, B. Liu, and H. Sun. Reduced spatio-temporal transcoding using an intra refresh technique. In *Proceedings of the International Symposium on Circuits and Systems (ISCAS)(4)*, pages 723–726, 2002.

[WNC87] I. Witten, R. Neal, and J. Cleary. Arithmetic coding for data compression. *Communications of the ACM*, 30:520–541, 1987.

[WSBL03] T. Wiegand, G.J. Sullivan, G. Bjntegaard, and A. Luthra. Overview of the H.264/AVC video coding standard. *IEEE Transactions on Circuits and Systems for Video Technology*, 13(7):560–576, July 2003.

[WZ98] Y. Wang and Q. Zhu. Error control and concealment for video communications: A review. *Proceedings of IEEE, special issue on Multimedia Signal Processing*, pages 974–997, 1998.

[XLS05] J. Xin, C-W. Lin, and M.T. Sun. Digital video transcoding. In *Proceedings of the IEEE (93)(1)*, pages 84–97, January 2005.

[YLZV04] X. Yang, J. Liu, F. Zhao, and N. H. Vaidya. A vehicle-to-vehicle communication protocol for cooperative collision warning. In *Proceedings of Mobile and Ubiquitous Systems: Networking and Services (MobiQuitous)(1)*, pages 114–123, August 2004.

[YS99] J. Youn and M.T. Sun. A fast motion vector composition method for temporal transcoding. In *Proceedings of the International Symposium on Circuits and Systems (ISCAS)(4)*, pages 243–246, May 1999.

[YSL98] J. Youn, M. T. Sun, and C. W. Lin. Motion estimation for high-performance transcoding. *IEEE Transactions on Consumer Electronics*, 44(3):649–658, August 1998.

[YSL99] J. Youn, M.T. Sun, and C.W. Lin. Motion vector refinement for high-performance transcoding. *IEEE Transactions on Multimedia*, 1(1):30–40, 1999.

[YSX99] J. Youn, M.T Sun, and J. Xin. Video transcoder architectures for Bit-Rate scaling of H.263 bit streams. In *Proceedings of the 7th ACM International Multimedia Conference (MM)*, pages 243–250, 1999.

[YVLS02] P. Yin, A. Vetro, B. Liu, and H. Sun. Drift compensation for reduced spatial resolution transcoding. *IEEE Transactions on Circuits and Systems for Video Technology*, 12(11):1009–1020, 2002.

[ZWC03] S. Zou, H. Wu, and S. Cheng. A new mechanism of transmitting MPEG-4 video in IEEE 802.11 wireless LAN with DCF. In *Proceedings of International Conference on Communication Technology (ICCT)(2)*, pages 1226–1229, 2003.

[ZZYW05] C. Zhang, S. Zheng, C. Yuan, and F. Wang. A novel low-complexity and high-performance frame-skipping transcoder in DCT domain. *IEEE Transactions on Consumer Electronics*, 51(4):1306–1312, 2005.