

ulm university universität
uulm

Video compression systems for low-latency applications

DISSERTATION

zur Erlangung des akademischen Grades eines

DOKTOR-INGENIEURS

(DR.-ING.)

der Fakultät für Ingenieurwissenschaften
und Informatik der Universität Ulm

von

RALF MICHAEL BEUSCHEL, GEB. SCHREIER

AUS FRIEDRICHSHAFEN

Gutachter:	Prof. Dr.-Ing. Albrecht Rothermel
	Mag. DI Dr. Margrit Gelautz, ao. Univ. Prof.
Amtierender Dekan:	Prof. Dr.-Ing. Michael Weber

Ulm, 22.01.2010

Video compression systems for low-latency applications

Abstract:

This dissertation presents an in-depth analysis of latency sources in real-time video compression systems and describes strategies to minimize the system delay.

Initially, the latencies of intra-frame and predictive coding in standard-mode video compression systems are examined. A theoretical analysis identifies the bitsream buffers – which are a pre-requisite of constant transmission bit-rate – as major sources of latency. In a second step, the results of experiments conducted using the "leaky-bucket" model on ten video sequences with sub-frame-level resolution are discussed. Based on these observations, the intra refresh coding method is identified as a key technique for low latency in predictive video compression systems. Finally, a novel approach for intra refresh coding is presented, which simultaneously satisfies the traditionally contradictory goals of high coding efficiency and low system latency.

Table of Contents

1 INTRODUCTION 1

1.1 Motivation 1

1.2 Background 1

1.3 Goals and methodology 3

1.4 Structure of the document 4

2 VIDEO COMPRESSION SYSTEMS 6

2.1 Physiological motivation of video compression techniques 6

2.2 Historical development of image and video compression standards 11

2.3 Technical aspects of video compression systems 13
2.3.1 Lossy image compression techniques 13
2.3.2 Principles of hybrid video codecs 15
2.3.3 Basic video coding tools 20
2.3.4 Temporal prediction tools 22
2.3.5 Compressed video stream hierarchy 25
2.3.6 Digital video representation 28

3 VIDEO CODING MODE ANALYSIS 32

3.1 Test Sequences 32

3.2 Definitions and quality metrics 34

3.3 Comparative coding mode analysis 35
3.3.1 Test set-up and quality adjustment 36
3.3.2 Intra coding mode analysis 38
3.3.3 IP-coding mode analysis 42
3.3.4 IPB coding mode analysis 47
3.3.5 Coding mode analysis summary 49

3.4 Localized data rate analysis 51
3.4.1 Macroblock data rate variations 51
3.4.2 Slice resolution and randomized data rate analysis 54

4 LATENCY ANALYSIS FOR STANDARD-MODE VIDEO COMPRESSION SYSTEMS 57

4.1 Generalized video transmission model 57

4.2 Algorithmic encoder and decoder latency 60

4.3 Transmission latency 63

4.4 System latency analysis for standard coding modes 67
4.4.1 Intra coding mode 67
4.4.2 Randomized intra coding mode 68
4.4.3 Predictive coding mode 69
4.4.4 Bi-directional prediction modes 71
4.4.5 Interlaced coding modes 73
4.4.6 System latency analysis summary 74

4.5 Experimental buffer latency analysis 76
4.5.1 Leaky bucket model 76
4.5.2 Intra coding mode 81
4.5.3 Predictive coding mode 84
4.5.4 Bi-directional prediction modes 87

4.6 System latency comparison for standard coding modes 88

5 OPTIMIZED INTRA REFRESH ALGORITHM 91

5.1 Introduction to the intra refresh principle 91

5.2 An adaptive region-based intra refresh scheme 95
5.2.1 Region-based forced intra refresh methods 95
5.2.2 Motion-adaptive intra refresh method 97
5.2.3 Selection algorithm for optimal refresh 101

5.3 Real-time implementation of the adaptive intra refresh 108
5.3.1 Frame-level flow chart 108
5.3.2 Structural overview 110
5.3.3 Estimation algorithm program flow 112

5.4 Intra refresh error recovery and scene changes 115

5.5 Low-latency rate control issues 117

5.6 H.264 Implementation aspects 121

5.7 Intra refresh coding mode latency analysis 124
5.7.1 Frame-CBR intra refresh 124
5.7.2 GOP-CBR intra refresh 128

5.8 Joint data rate and latency optimization 133

5.9 Coding mode performance summary 137
5.9.1 Buffer latency comparison 137
5.9.2 System latency summary 138
5.9.3 Latency, data rate and error recovery comparison chart 139

6 IMPLEMENTATION CONSIDERATIONS FOR LOW-LATENCY VIDEO COMPRESSION SYSTEMS 143

6.1 Introduction 143

6.2 DSP implementation of an MPEG-2 encoder 144
6.2.1 Detailed functional overview 144
6.2.2 Program-flow optimizations for low-latency encoders 145
6.2.3 Memory usage and access optimizations 147

7 SUMMARY & CONCLUSIONS 152

8 BIBLIOGRAPHY 154

9 LIST OF ABBREVIATIONS 162

10 LIST OF SYMBOLS 165

11 LIST OF FIGURES 166

12 LIST OF TABLES 169

APPENDIX A – H.264 AND MPEG-2 QUALITY ADJUSTMENT 171

APPENDIX B – SIMULATION RESULTS 173

APPENDIX C – INTRA REFRESH ALGORITHM FLOWCHARTS 176

APPENDIX D – INTRA REFRESH EXCESS DATA RATE SIMULATIONS 178

APPENDIX E – INTRA REFRESH TABLE INITIALIZATIONS 180

APPENDIX F – ENCODER CONFIGURATION 186

1 INTRODUCTION

1.1 Motivation

Controlling and minimizing the latency of real-time video compression systems is becoming more and more critical as this technology is employed in applications which traditionally use uncompressed, latency-free video transmission. With a focus on video broadcast and video conferencing, contemporary video compression standards were not designed to meet the latency requirements of emerging applications such as automotive driver assistance cameras, remote controls or RF studio cameras. Especially if high compression ratios must be achieved, the common techniques introduce unacceptable buffer delays to achieve a constant transmission bit rate. Therefore these new applications can only be realized if the compression tools and algorithms as well as the target hardware are optimized for minimum latency.

Designing systems for low-latency video transmission requires basic implementation knowledge as well as profound knowledge of the video coding algorithms and buffer management. Video compression standards only describe the algorithmic decoding procedure and profile features which ensure the interoperability of different encoders and decoders. Nevertheless, the system designer has many options in selecting coding modes, coding parameters and buffer sizes to meet the requirements of a specific target application.

1.2 Background

Modern video coding applications are based on the concept of hybrid video coding, which is a combination of lossy intra picture coding and (principally lossless) inter picture prediction. As bandwidth was not a major concern in classical video broadcast applications, the video compression technology was initially driven by video conferencing applications. The enabling technology for efficient video compression is high-performance digital signal processing, because improvements in the coding efficiency are mostly achieved at the cost of increased computational complexity.

Since the first digital video coding standard released in 1984, it has been possible to improve the efficiency continuously because of the decreasing cost of computing performance. Furthermore, the progress in storage and communication technology has enabled higher data rates, allowing to continuously increase the image resolution and visual quality over the last 20 years.

In the course of the last two decades the video coding technology was mainly driven by four applications, namely:

- video conferencing and video telephony
- video storage and retrieval, e.g. video archives, video-CD, DVD (digital versatile disc)
- video broadcast, e.g. DVB (digital video broadcast), DMB (digital media broadcast)
- (Internet) streaming

Due to the different requirements of these applications, a significant number of standardized and proprietary video coding techniques were established. Furthermore, within all relevant standards, the coding parameters can be adjusted to match the characteristics of different applications.

For the purpose of storage and retrieval, the compression efficiency is usually the major concern to achieve the highest possible quality with a minimum storage cost. Hence, these systems use the most advanced algorithms which require the highest computational performance, backward temporal prediction and possibly a pre-analysis of the complete video clip. This is, however, not a critical issue if the encoding can be done offline without real-time constraints of resources and algorithms. The characteristics of archived video streams are generally contradictory to low-latency transmission and processing.

Live broadcasting and streaming applications impose certain real-time constraints on the system which limit the choice of algorithms as it is not possible to perform a pre-analysis of the complete video. In the case of video broadcast servers, the computational complexity is a minor issue since initial hardware costs pay off quickly against broadcast transmission bandwidth. The high coding efficiency of broadcast applications is achieved with the same coding tools as used for storage applications but with a short pre-analysis interval of a few frames. This results in processing, buffer and transmission delays in the range of multiple seconds.

In the case of video conferencing and video telephony applications, the requirements differ significantly. The major issue here is the reduction of latency to an end-to-end delay of $100\ ms$ which is required for unimpaired conversation. Therefore, the video transmission must achieve the same delay to guarantee the lip-sync of audio and video. Limitations on computational resources in mobile and relatively cheap consumer devices also restrict the choice of algorithmic tools resulting in reduced coding efficiency. Furthermore, the point-to-point characteristics of video conferencing impose restrictions on the bandwidth because the transmission capacity is paid by only two users rather than a large broadcasting audience.

1.3 Goals and methodology

It is the goal of this thesis to identify the potential of existing video coding technology in applications which require very low latency video transmission. Due to the lack of optimized video compression systems, presently many applications can only be realized with uncompressed video transmission to fulfill the latency requirements. It is therefore the intention of this thesis to give guidelines for the delay estimation of complete video transmission systems and provide algorithmic tools to optimize the coding efficiency under low-latency constraints.

Beyond the classical video conferencing scenario, new applications such as interactive remote control systems demand latencies which cannot be achieved with conventional frame resolution buffer models and rate control schemes. The focus of this work therefore lies in the development of coding methods which allow system latencies significantly below $100\ ms$ while maintaining the highest achievable coding efficiency.

Taking into account the different bandwidth requirements of applications, simple methods like intra coding as well as complex coding methods using temporal prediction are analyzed. In the case of automotive applications, an intra-frame compression may be sufficient for transmitting high resolution images over standard resolution infrastructure. However, if multiple trailer cameras for parking-aid use a common wireless transmission channel, maximum compression efficiency may be a more important factor than extremely low delay.

Concerning low-latency applications, the available literature mainly discusses rate control, frame-skipping and network optimizations with a focus on video telephony and variable bitrate (VBR) streaming applications. It is, however, not the goal of this thesis to develop or evaluate a specific algorithm of this kind. In contrast to this, the approach taken in this thesis is to analyze the inherent system behavior without rate control means under the constraints of constant bitrate (CBR) transmission mode. Based on the delay analysis, modifications of the encoding scheme are proposed for delay reduction. With the described methodology, operating bounds and requirements for the rate control and buffer management algorithms are determined based on a set of representative video test sequences.

Especially in safety sensitive applications, the re-synchronization time in case of transmission errors becomes a major design criterion. In contrast to uncompressed video transmission, highly efficient temporal prediction can introduce error propagation effects from which the system needs to recover within a pre-defined time interval. Selecting the coding parameters for low-latency applications is therefore mainly a tradeoff between bandwidth requirements, limitations on rate fluctuations, re-synchronization capability and computational requirements. The

analysis in this document evaluates these parameters for all relevant coding modes in broadcast or point-to-point applications with a guaranteed re-synchronization time.

1.4 Structure of the document

In this section a brief introduction to the structure of the document is given along with the high-level concepts of the underlying research.

The document starts with an introduction to video compression technology in chapter 2. A discussion of the relationship between human visual perception and video data reduction techniques introduces the sequence of basic processing steps, which is typical for all image capture and compression systems. This discussion can be seen as a historic development from chemical still-image cameras to advanced digital compression techniques. In section 2.2 the evolution of digital video compression techniques in the last 20 years is described in the context of the standardization activities. Section 2.3 summarizes the concepts of lossy image compression techniques and hybrid video codecs, which form the basis of all relevant video compression standards.

In chapter 3 an analysis of H.264 and MPEG-2 compression tools is presented based on a set of ten standard video test sequences whose characteristics are described in section 3.1. The simulation results presented in section 3.3 include a detailed performance evaluation of temporal prediction tools which form the basis for subsequent discussions. A second major outcome of these simulations is the analysis of data rate variations with different levels of temporal resolution. While section 3.3 focuses on the frame-level data rate variations, section 3.4 gives an insight on the sub-frame rate variations down to the smallest processing units.

Based on these observations, an inherent CBR interval for each coding mode is defined, which is used for the subsequent latency analysis in chapter 4. Section 4.1 introduces a generalized latency model for compressed video transmission systems which distinguishes algorithmic and transmission related latencies. The sources of algorithmic delays are discussed in section 4.2 and the latency optimization potential which can be achieved by improved process-scheduling is shown. Section 4.3 gives a first introduction to the buffer and transmission latencies caused by the constraints of constant bitrate video transmission.

After these introductory sections, section 4.4 gives a system level latency analysis for the commonly used coding modes with consideration of buffer and algorithmic delays. Simple buffer delay approximations are derived based on the inherent CBR interval, which allows the identification of the potential for low-latency video transmission. As the conventional frame-based buffer modeling approach does not fulfill the desired very low delay requirements, the results of an experimental

buffer analysis considering sub-frame data rate variations are presented in section 4.5. With detailed buffer simulations for the selected ten test sequences, average and maximum buffer requirements for each coding mode can be derived and the results are summarized in section 4.6.

The results of chapter 4 clearly indicate that the desired system latencies below $100\ ms$ cannot be achieved with standard predictive coding modes. Therefore, the basic intra refresh method is introduced in section 5.1 which allows low delays and high temporal prediction gains at the same time. Section 5.2 focuses on the forced intra refresh method which guarantees the error recovery within a defined time interval. In sections 5.2.2, 5.2.3 and 5.3 an innovative algorithm is presented which adapts the forced intra refresh strategy to the motion of the video sequence. In this way, the degradation of coding efficiency can be avoided as it would occur if static intra refresh strategies are used. The following sections focus on details of the intra refresh like error recovery, rate control and H.264-specific issues.

In the subsequent section 5.7, the experimental latency analysis for the intra refresh coding modes is performed and in section 5.8 the feasibility of joint optimization for data rate and latency is shown. In section 5.9 these results are put into context with the system level analysis and the coding mode data rate evaluation of section 3.3. The chapter is completed with a joint consideration of delay, data rate and error recovery performance which allows quick selection of appropriate coding modes and parameters based on the application requirements.

In chapter 6 the implementation of a low latency MPEG-2 encoder on a programmable digital signal processor (DSP) is described. This “proof-of-concept” implementation forms the basis for the algorithmic delay optimization discussion in section 4.2. Chapter 7 summarizes the results of this document.

2 VIDEO COMPRESSION SYSTEMS

2.1 Physiological motivation of video compression techniques

The development of video compression did not just start with the MPEG video standards, it can be seen as an evolutional process since the invention of photography. In this development over the last 150 years a correspondence between shortcomings of the human visual system and technical benefits can be seen. At the beginning, video transmission was of course limitation-driven: when the first movies were produced with chemical film it was just not possible to capture color. So, taking just the black and white images seemed a reasonable compromise because the viewer could obviously still recognize actors and objects. Later developments required however deeper knowledge of the visual processing of the human eye and brain, also referred as the "human visual system" (HVS), before they could be applied in the image processing system. A major step in this direction was the fundamental research of human color vision for color television systems undertaken by the CIE (Commission Internationale de l'Éclairage) in 1931.

Video compression techniques can be divided into two major groups [1]:

- **Removal of redundant information**
  Redundancy is information which can be discarded without loss of information. This can be entropy redundancy as used in the context of information theory as well as spatial and temporal redundancy which is specific to image and video coding.

- **Removal of irrelevant information**
  Irrelevancy in the context of image and video compression can be defined as information that the human viewer does not notice if it is missing or modified. It is therefore related to the way the human visual system works and exploits the shortcomings of the human visual system. In some publications irrelevancy is also referred to as psycho-visual redundancy.

The underlying problem of lossy image and video compression is to define the irrelevancy and to remove it gently enough with minimum visible coding artifacts.

In the following section the fundamental processing steps of current video compression systems are summarized and the relationship with the physiological properties of the HVS is explained. The sequence of processing steps as postulated in table 2.1 is sorted in the order of the technical system implementation and references are given to the relevant literature in the discussion.

Table 2.1: Relationship of lossy video compression techniques and the properties of the human visual system (HVS).

	Technical system implementation	**Property of the HVS**
1.	3-D space is projected to 2-D image	The human brain can mostly reconstruct the spatial composition of scenes from overlapping, light, shade, motion and experience.
2.	Sampling with about 25 images/s, displaying with at least 50 images/s to avoid flicker.	The HVS accepts low temporal resolution for low-motion video but it is very sensitive to flicker.
3.	Projection of visible light spectrum to three color primaries, ideally representing linear combinations of the human spectral sensitivity curves.	The eye has three types of cones for color vision with different spectral sensitivities.
4.	Scanning of images in horizontal and vertical direction	The eye has a limited spatial resolution.
5.	Non-linear mapping of input luminance values (gamma correction / signal compression towards high brightness levels).	The eye has a non-linear sensitivity for intensity differences.
6.	Conversion to YUV color space.	The retina generates luminance and chrominance information from the three receptor stimuli.
7.	Reduction of color resolution in U and V channels.	Reduced spatial sensitivity for chrominance information is well tolerated.
8.	Standard display and camera technology is limited to $50\,dB \sim 60\,dB$ dynamic range. High dynamic range cameras allow up to $140\,dB$.	The sensitivity of the eye covers a dynamic range of $280\,dB$ by adaptation mechanisms. Within a scene, the dynamic range is strongly limited.
9.	Reduced resolution for higher spatial frequencies	The contrast sensitivity function of the HVS decreases towards higher spatial frequencies.
10.	Exploit correlation of neighboring pixels (spatial redundancy, lossless)	
11.	Exploit similarities between successive images (temporal redundancy, lossless) .	

According to this scenario, capturing and processing of video for electronic storage and retrieval involves up to eleven processing stages where steps one through nine remove irrelevant information (psycho-visual redundancy), and only steps ten and eleven can be realized as redundancy removal without information loss. Additionally, entropy coding techniques can be applied prior to storage or transmission of the video.

Step 1: Capturing images and moving pictures is usually achieved with a single camera resulting in a loss of depth information which is mostly compensated by monocular and binocular cues of the HVS [2]. From the analysis of the scene, e.g. overlaps and size ratios, the human brain can reconstruct the spatial composition of a scene. For special applications like remote controls, stereoscopic cameras can capture a more realistic representation of a 3-dimensional scene for a certain viewing position but the display technology is still the weak point of this technique for multi-viewer applications.

Step 2: For the reproduction of smooth motion a constant number of pictures per time interval is sampled which generates the impression of true motion in the brain if the reproduction is done accordingly [3]. The visual system integrates temporal information but the required minimum "flicker rate" of flashing reproduction techniques (i.e. cathode-ray tube (CRT) or film projection) depends on the overall luminance of the image and the viewing angle [4]. The flicker sensitivity of the human eye ranges from about $55\ Hz$ at $20\ cd/m^2$ luminance at 20 degrees viewing angle to about $85\ Hz$ for $220\ cd/m^2$ at viewing angles over 60 degrees.

The impression smooth motion is not directly related to the "flicker rate" as various techniques (interlace TV, repeated picture flashing) can be used to increase the flicker rate even if a lower picture frame rate is used. Traditionally movies on chemical film are recorded with $24\ fps$ (frames per second) which is satisfactory for slow and medium motion activity. For cartoon productions the frame rate is $12\ fps$ but it is frequently increased to $24\ fps$ if quick motion is displayed [5].

Step 3: The retina of the eye has three types of cones (L, M, S) which are sensitive to different parts of the visible light spectrum. Hence, the eye provides a weighted summation of the different spectral components rather than processing detailed spectral information [6,7]. It is therefore possible to generate the same color impression with different spectral composition of the light (metamerism) and it can be described with a three-dimensional space of three color primaries. In the technical system the visible light spectrum is projected to three color primaries, ideally representing linear combinations of the human spectral sensitivity curves.

Current display technology however does not cover the complete human color space and it is especially limited in the range of highly saturated colors.

Step 4: In the case of electronic reproduction with analog components, the image is scanned in a line-wise manner resulting in sampling of the vertical direction. For digital image processing a second sampling step must be applied in the horizontal dimension resulting in an orthogonal sampling grid of small picture elements (pixels). Viewing a scanned image therefore requires a certain minimum viewing distance such that the viewer does not notice the scanline or pixel structure of the display. The acuity of the human eye is about 50 cycles per degree at high luminance levels ($\sim 1000\ cd/m^2$)[8]. Considering the image resolution of standard resolution television (576 lines), the minimum viewing distance should be larger than 4.6 times the image height.

Step 5: In early TV systems a gamma correction on the transmitter side was introduced to compensate for the non-linear characteristic of the picture tube. As the human eye has an overall logarithmic characteristic, it is more sensitive to luminance variations in dark image regions rather than in bright image regions. Therefore, if a CIE-like $L^* \sim L^{0.4}$ gamma correction is applied to the input values, the resulting lightness expands low luminance levels and compresses high luminance levels [9]. This results in better visual image quality because amplitude errors in critical dark image regions are reduced.

As contrast sensitivity of the human eye depends on the absolute luminance level, the same change of relative luminance appears different in magnitude for the viewer. The DICOM [10] standard for medical imaging introduced a "perceptually linearized" display function using JND units (just noticeable difference) which define the smallest detectable luminance difference at a certain luminance level. For high dynamic range imaging logarithmic rather than square root correction functions can be used.

Step 6: The retina transforms the three cone stimuli into an achromatic channel and two difference signals which code the red-green and blue-yellow color differences [6]. The YUV / YCbCr color space used for image transmission and compression was defined in such a way that a signal change on the luminance (Y) channel results in zero activity of the chromatic channels of the HVS.

Step 7: The experimental results presented in [11] indicate that the red-green difference signal (Cr or V component) has about half the spatial bandwidth of the achromatic channel. The spatial bandwidth of the blue-yellow (Cb or U) channel is assumed to be limited by two factors, the chromatic aberration of the eye and the sparse density of S-cones sensitive to the blue light spectrum.

The chromatic aberration leads to a defocused vision in the blue spectrum because the lens is normally focused for maximum resolution of the L and M cones light spectrum resulting in highest spatial resolution of the achromatic channel. As the blue color spectrum cannot be focused at the same time, significant spatial resolution is lost in the blue-yellow channel.

Step 8: The HVS covers a luminance range from 10^{-6} to 10^{8} cd/m^2 (280 dB) which is achieved by adaptation of the iris and the retina [8,9]. The photopic vision of the cones relevant for color image reproduction ranges from 10^{-2} to 10^{8} cd/m^2 (200 dB), the scotopic vision of the rods from 10^{-6} to 10 cd/m^2. For office environments the required illumination is in the range of 100 lux to 1000 lux, resulting in roughly 25 to 250 cd/m^2 luminance for white paper. Current LCD (liquid crystal display) technology can generate about 300 cd/m^2 luminance and contrast ratios of up to $1400{:}1$ (63 dB).

For current display technologies with approximately 300 cd/m^2 luminance, the eye can distinguish about 630 JND brightness levels with reference to a 0.05 cd/m^2 black level of the display [10]. As the contrast ratio limits the minimum brightness of current LCDs at 0.2 cd/m^2 $= 300$ cd/m^2 $\bullet$ $1/1400$, about 600 JND levels would be required for contouring-free grayscale displays if the DICOM standard is used for storage and interfacing. In both cases a nine to ten bit representation of the source signal would be required which is usually not the case in current consumer electronic products.

Using HDRC (high dynamic range CMOS) sensor technology and evolving displays with a luminance range of 0.01 to 10000 cd/m^2 [12], a logarithmic quantization scheme should be used with a resolution of 10 to 11 bits. The necessary modifications could be incorporated into existing video compression standards as discussed in [13].

Step 9: The contrast sensitivity function decreases towards higher spatial frequencies, allowing quantization with less precision for higher spatial frequencies [14,15].

Step 10 & Step 11 are implemented as lossless coding tools in current video standards, i.e. spatial intra prediction (step 10) and temporal prediction (step 11). Related features of the HVS could be seen in higher level image segmentation and motion tracking processing.

Concerning video transmission and compression techniques it can be concluded that analog color TV technology already employed the processing steps 1 to 8 to achieve acceptable visual quality with the available bandwidth. A lossy intra frame image compression algorithm would add processing step 9 to reduce the amount of data at the cost of small quantization errors. Spatial and temporal

prediction tools (step 10 and 11) were introduced with the first video compression standards and rely heavily on computational expensive algorithms.

2.2 Historical development of image and video compression standards

In this section a short summary of the video compression standards development of the last two decades is given and illustrated in figure 2.1. A detailed overview of the historic development of video compression standards and their technical details is given in [16].

The fundamental concepts of current video coding technology were developed in the 1950s to 1980s and first applied in the digital video coding standard H.120 which was released in 1984. The core algorithms were differential pulscode modulation (DPCM) coding and conditional replenishment which are not used any more in today's video codecs (encoding/decoding device). In a second release of 1988 motion-compensated predictive coding was added to this standard but the development in DPCM coding ceased when modern hybrid video coding systems emerged.

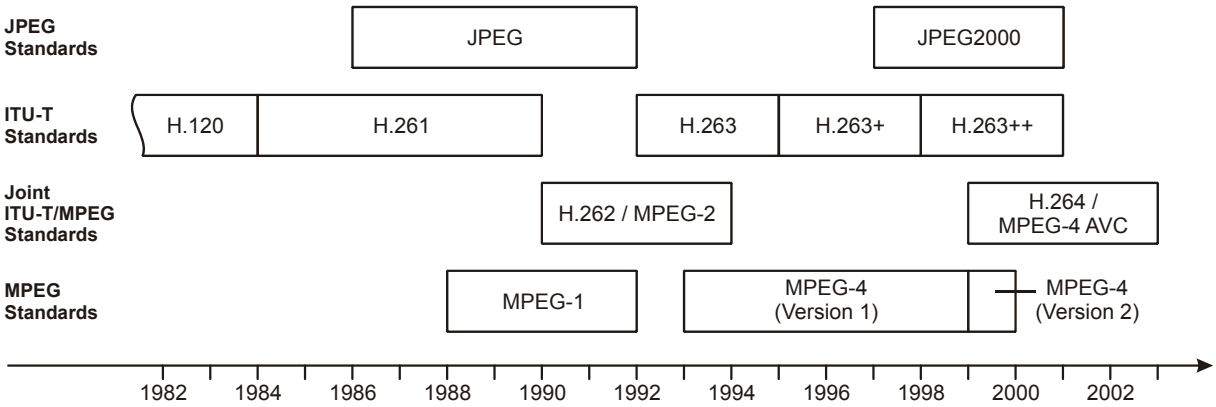


Figure 2.1: Historical development of video standards.

Current video codecs are conceptually based on the H.261 video conferencing standard which was started in 1984 and released in 1990 by the ITU (international telecommunications union) standardization organization. It includes the key features of macroblock-based motion compensation, transform coding, entropy coding as well as an hierarchical stream structure and it was widely used in video conferencing applications. At the same time the JPEG still image coding standard was developed by the "Joint Photographic Experts Group" of ITU and ISO (international standardization organization). After evaluating several coding techniques, a DCT (discrete cosine transformation) based method was selected in 1988 as basic compression algorithm [17]. JPEG and H.261 as well as succeeding video standards use very similar coding techniques based on the 8 by 8 pixel 2D DCT block transformation.

The MPEG-1 video standard (ISO/IEC 11172-2), a direct successor of H.261, was developed from 1988 to 1992 by the ISO/IEC standardization targeting video storage on CD-ROMs at 1.15 *Mbit/s* data rate. It inherited the basic concepts and uses the same coding tools as H.261 but extensions and modifications were made for offline storage and higher flexibility. A major improvement was however the introduction of bi-directional prediction, allowing higher compression at the cost of increased processing delay which is usually undesirable in video conferencing applications. The image size was still limited to quarter standard TV resolution with progressive scan. In addition to the basic video compression algorithm, the standard also includes audio compression tools like the famous MP3 codec as well as audio-video multiplexing tools.

From 1990 to 1994 the MPEG-2 standard was developed as a joint project of ISO and ITU working groups and the video profile was approved as ISO/IEC 13818-3 [18] and ITU-T H.262 standard. It is a superset of MPEG-1 video and forward compatibility is required for MPEG-2 decoders to support decoding of MPEG-1 streams. Major improvements in the video coding part include efficient coding of interlaced video, scalability, higher image resolution and bit rate profiles for high definition video. The standard became a great success as it was adopted in many consumer products like DVDs for video storage and DVB.

The development of the ITU video conferencing standards continued with the H.263 standard from 1992 to 1995 and two enhanced versions were released in 1998 (H.263+) and 2001 (H.263++). H.263 and its successors became the dominant video conferencing codecs as their efficiency is optimized towards low bitrate applications.

MPEG-4 was designed as a standard describing scenes comprising various media sources. Therefore, video coding tools for arbitrarily shaped objects, scene composition tools as well as still image modification tools were included in this standard. The video codec MPEG-4 part 2 (ISO/IEC 14496-2) was released in 1999 and it is based on the H.263 baseline codec with additional coding tools for improved performance. Two enhanced versions of the video codec were added in 2000 and 2001. The advanced simple profile (ASP) MPEG-4 video coder became famous as DivX codec but it was not as successful in commercial products as MPEG-2. This may be due to the fact that the coding efficiency was not dramatically improved compared with a highly optimized implementation of the MPEG-2 codec to justify the significantly higher licensing fees.

In 1997 the development of the JPEG2000 still image compression standard was started which was officially released in December 2000. It is the first image compression standard which employs the wavelet transformation resulting in improved image quality especially at high compression ratios. The intra frame Motion-JPEG2000 codec is targeted for applications where still images as well as

image sequences are processed but it does not achieve the efficiency of predictive video coding methods. Nevertheless, JPEG2000 was officially adopted for the distribution of digital video content for digital cinemas by the DCI organization (Digital Cinema Initiative) [19].

Standardization work on H.264 (initially named H.26L), a joint project of ITU and ISO, started in 1999 and was completed in 2003. In contrast to previous ISO standards it describes only a video codec and does not provide stream multiplexing or audio coding features, hence it was officially adopted as ISO/IEC 14496-10, alias MPEG-4 Part 10 or MPEG-4/AVC. Its primary development targets were significant improvements in coding efficiency, error robustness, exact match decoding and enhanced quality for low latency applications.

From the implementation point of view MPEG-2 is the standard with smallest computational requirements offering all features required for interlaced video transmission. It was therefore chosen for the real-time implementation of the low-latency video encoder as described in chapter 5.9.

All named video standards use a combination of DCT-based irrelevance reduction and temporal prediction. The one-sided coding scheme results in a significantly more complex encoder with a simple decoder, targeting mass markets with cheap decoding hardware and more expensive studio equipment.

All discussed standards describe the decoding process and profile-dependent restrictions on the data streams. Furthermore, reference implementations of the video encoders and decoders developed by the standardization committees are usually available. In the case of MPEG-2, the final released version was the TM5 (Test Model 5, [20]). As the reference implementation H.264 was still under development in the course of this thesis, the experiments presented in this document were carried out with the JM7.5b version [21].

2.3 Technical aspects of video compression systems

2.3.1 Lossy image compression techniques

Lossy image and video compression techniques play a dominant role in current multimedia consumer products as they allow cost-efficient storage and transportation of visual content. In contrast to lossless compression techniques, the lossy image compression allows small errors in the image representation compared with the uncompressed original image which may be more or less invisible for the viewer. An overview of fundamental image compression techniques is given in [16,22,23,24].

The generic lossy image compression system is shown in figure 2.2. In a first step, the image is split into independent rectangular blocks on which a 2-dimensional transformation is applied resulting in a spectral decomposition of the input image. The analysis of natural images reveals a high spatial correlation of neighboring pixels. It is therefore the intention of the transformation stage to remove the spatial correlation and concentrate the relevant information in a small number of significant transform coefficients.

The spectral decomposition also prepares the input data for the succeeding frequency adaptive quantization which considers the spatial frequency sensitivity characteristics of the human visual system. In the last step, the output of the quantizer is entropy encoded using common techniques like variable-length coding (VLC), e.g. a combination of run-length encoding (RLE) and Huffman coding, or binary coding schemes. As the output of the quantizer has a lower entropy than the input image, the transformed and quantized image can be compressed with less bits than the original image.

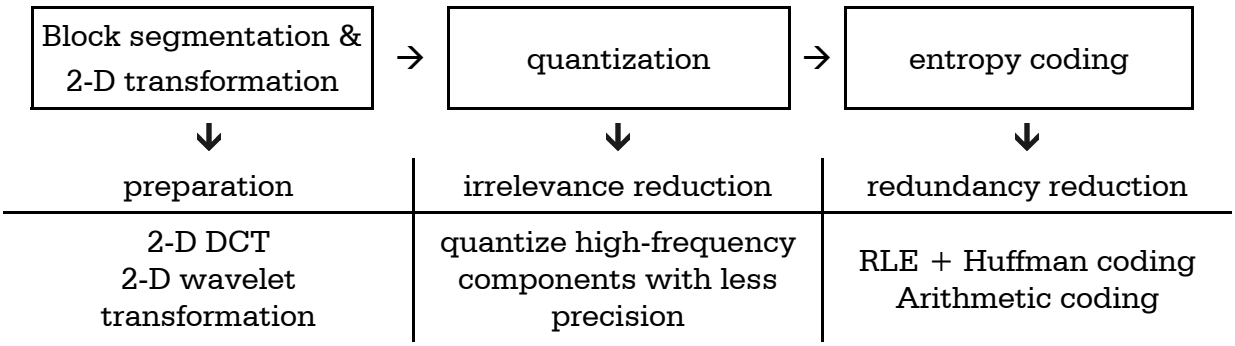


Figure 2.2: Lossy image compression techniques.

The real-valued discrete cosine transformation (DCT) is used in many standards, e.g. JPEG, MPEG-1, MPEG-2, DVCAM [25], because it offers a good trade off between computational complexity and coding efficiency. It is defined by the block size N and the following equation, where x, y represent coordinates in the spatial sample domain and u, v represent coordinates in the transform domain [18]:

$$F(u,v) = \frac{2}{N} C(u)\, C(v) \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f(x,y) \cos\frac{(2x+1)u\pi}{2N} \cos\frac{(2y+1)v\pi}{2N}$$

$$with \qquad u,v,x,y \in \{0,1,2,\ldots N-1\} \tag{2.1}$$

$$where \qquad C(u), C(v) = \begin{cases} 1/\sqrt{2} & for\ u,v = 0 \\ 1 & otherwise \end{cases}$$

Increasing the block size N results in better coding performance but higher computational complexity growing with the square of N. The DCT block size of N=8 pixels is used by many common standards as a tradeoff between coding efficiency, complexity and convenience of implementation. The DCT requires however fractional multiplications which can lead to precision problems in fixed-

point implementations. In newer standards these precision problems are solved by introducing numerically less-critical integer transformations, e.g. H.264, AVS, WM9 [26,27,28].

In recent years the wavelet transformations were introduced for image compression, e.g. the JPEG2000 standard [17], which are based on subband decomposition of the input image. The coding algorithm uses an iterative two dimensional lowpass-highpass filtering method followed by subsampling of the band-limited images. This principle allows generation of low-resolution images by truncation of the data stream if the data is organized accordingly.

The quantization step is realized by dividing each transform coefficient with a frequency-dependent constant. These constant values are stored in the quantization tables which can be standard or user defined and known to the encoder and decoder. Furthermore, a second divisor can be introduced for adjusting the quantization precision within the image. As a result of the quantization process, many transform coefficients are quantized to zero because the energy is concentrated in a few significant transform coefficients. For the implementation of a good rate control algorithm it is essential to model the relationship between the data rate and the quantizer value.

2.3.2 Principles of hybrid video codecs

Codec architecture

Figure 2.3 illustrates the structure of a hybrid video transmission system [1] ("codec"), consisting of a video encoder and a matching decoder. The inner encoder is formed by the 2-D transformation, quantization and entropy encoder which is identical to the structure of a lossy still-image encoder. For exploiting the temporal correlation between successive frames, a prediction formed from the previous frame is subtracted from the current input video signal prior to the transformation stage. For achieving a good coding performance it is necessary to form the temporal prediction considering the motion that occurred between the frames.

Current video codecs generally use block segmentation and rectangular transformation blocks. Transformation block sizes vary from 8 by 8 pixels in the older coding standards to 4 by 4 in H.264. The temporal prediction of current video codecs is based on rectangular blocks of a size between 4 by 4 and 16 by 16 pixels. Hence, one or more basic DCT blocks are grouped and transmitted together with an associated motion vector describing the spatial offset from the reference image.

The motion estimator finds the best matching reference region for the currently processed frame segment and passes the calculated offsets (motion vectors) to the prediction block. The temporal prediction uses the previously coded and reconstructed frames as reference, because the original input image is not available in the decoder. Coding experiments presented in [29,32] show that the motion-compensated prediction generally outperforms the temporal prediction without motion compensation (using only zero motion vectors) significantly.

The feedback loop of the encoder includes the major functional blocks inverse quantization and inverse transformation of the decoder in the reconstruction unit. Furthermore, it uses exactly the same reference and error signals for the temporal prediction as the decoder if no transmission errors occurred. However, the structure of the temporal prediction loop in the decoder is similar to an IIR (infinite impulse response) filter structure, which results in an infinite error accumulation and error propagation in temporally predicted frames. As a consequence, the hybrid video transmission is quite susceptible to transmission errors and mechanisms for re-synchronizing the encoder and decoder predictors are required.

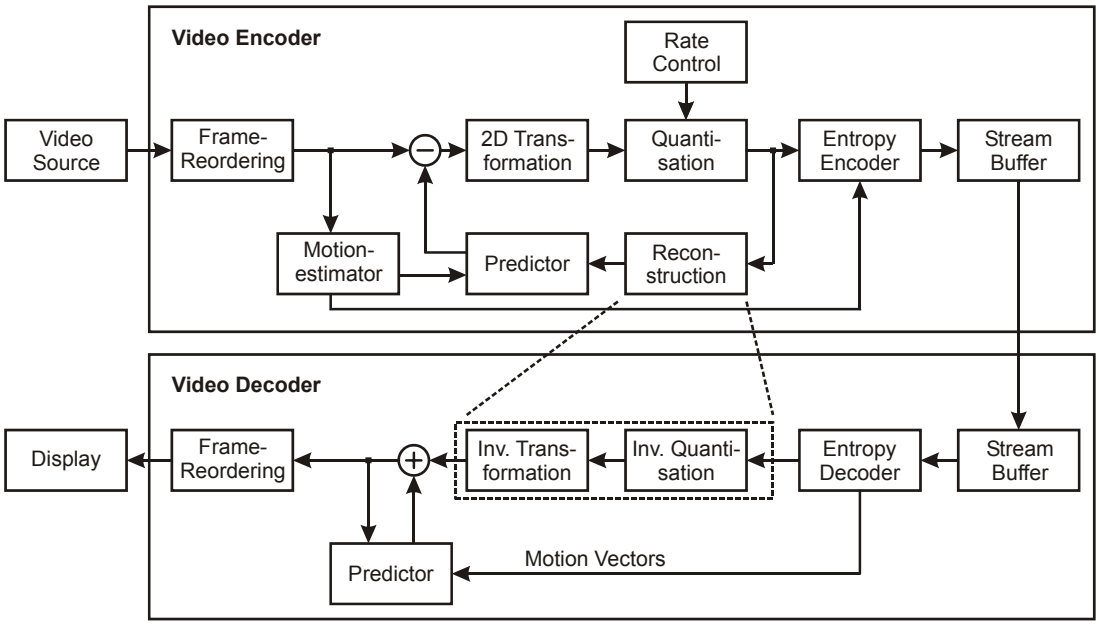


Figure 2.3: Hybrid video codec transmission system.

Temporal prediction modes

Using the above hybrid video codec, all common video coding standards offer three ways of coding a frame:

- **Intra-coded frames (I-frame)**
  I-frames are coded without temporal prediction allowing instantaneous decoding without knowledge of previous frames. They are intended as starting

points for decoding the video stream and also enable trick modes like fast forward and backward skipping.

Intra-coded frames use only lossy image compression in spatial domain similar to conventional DCT based image compression tools like JPEG. Commonly, an I-frame is inserted about every $500\ ms$ in the compressed data stream for eliminating possible drift between the encoder and decoder which may be caused by transmission errors or numerical inaccuracies.

- **Predicted frames (P-frames)**
  P-frames are generally predicted from one or more previous frames, most frequently a directly preceding I-frame or P-frame in the data stream as shown in figure 2.4. In IP-coding mode, the video encoder usually inserts a constant number of N_P predicted frames between two I-frames. The remaining error after subtracting the motion compensated prediction from the current image is transformed using the DCT and quantized. For common video sequences, P-frames achieve a rate reduction of $50\,\%$ to $90\,\%$ compared with I-frames. The coding efficiency of predicted frames can be increased by allowing more than one past reference frames and a flexible reference selection on macroblock level.

- **Bi-directional predicted frames (B-frames)**
  B-frames are predicted from the two surrounding reference frames which can be any combination of I and P-frames. The prediction in IPB coding mode can be formed by using only the past or future reference frame prediction, or by averaging the two predictions.

  B-frames allow the highest compression ratio resulting in a roughly $30\,\%$ to $80\,\%$ reduced data rate compared with P-frames. The coding gain depends however on the complexity of motion in the video sequence because the increased distance of the P-frames results in a higher data rate for these frames. Hence, under certain circumstances the coding gain of the B-frames can be compensated completely by the P-frames as indicated by the simulations in section 3.3.4. Other drawbacks of B-frame coding are an increased computational complexity, additional frame reordering memories and an increased latency in real-time transmission systems. Low-complexity profiles of video standards, e.g. MPEG-2 simple profile (MPEG-2 SP) and H.264 baseline profile (BP), limit the functionality to I and P-frames, avoiding expensive processing resources for B-frames.

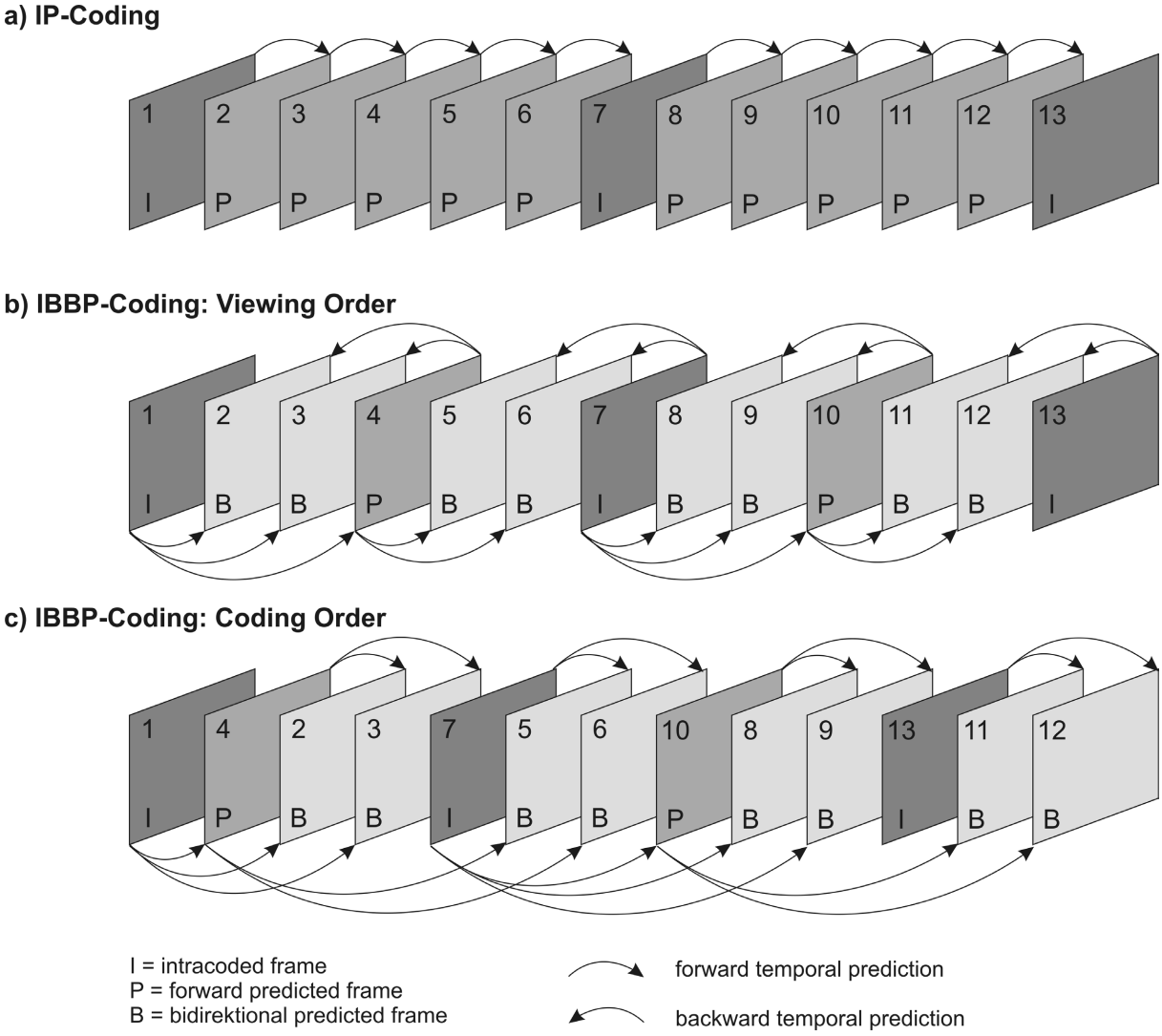


Figure 2.4: Temporal prediction with IP and IPB coding schemes.

It should be noted that parts of the frame are sometimes intra-coded within a P or B-frame if the motion estimation fails. It is also possible to embed intra-coded regions into predicted frames to adapt the codec for special applications.

The newer standards H.264 [30] and AVS [31] introduce additional spatial intra prediction modes to improve the coding efficiency. The intra prediction tries to predict the current macroblock data in spatial domain by replicating texture information from already transmitted neighboring macroblocks to the current macroblock position. If a good texture prediction is found, only the remaining residual is transmitted rather than the complete intracoded information.

Standard GOP structures

The GOP-structure (group of pictures structure) characterizes the sequence of I, P and B-frames in the data stream generated by the video encoder. As it is not fixed within the standard it can be adapted to the requirements of the application and

the video material. However, some applications like DVDs, DVB or DMB impose restrictions on the maximum distance of I-frames.

Throughout this document the naming conventions as defined in table 2.2 are used for the different coding modes. Furthermore, the following definitions apply for the GOP-structure for IP and IPB encoded video streams:

$$T_{GOP} = N_{GOP} \bullet T_{frame} = (1 + N_P + N_B) \bullet T_{frame}, \quad \text{with} \tag{2.2}$$

T_{GOP} = GOP period
T_{frame} = frame period
N_{GOP} = number of frames in GOP
N_P = number of P-frames in GOP
N_B = number of B-frames in GOP

Intra-coded frames allow instantaneous decoding start without knowledge of prior frames. I-frame only coding is therefore commonly used for studio and video editing applications where random access to arbitrary frames is required. This mode results in a simple encoder without motion estimator, inverse transformation and reference picture storage. The coding efficiency is however fairly poor.

IP-coding inserts a constant number of P-frames, usually 12 to 15, between the I-frames. Increasing the amount of P-frames results in a better coding efficiency but the error propagation caused by transmission errors or numeric inaccuracy also increases. Compared with the I-frame only encoder, the IP-encoder requires additional IDCT (inverse DCT), picture reconstruction and motion-estimator function blocks as well as one reference frame memory.

The highest compression can be achieved using IPB-coding if the video has low or medium motion. Historically B-frames were exclusively predicted from the surrounding I and / or P-frames and never served as reference frames. With this limitation the complexity of the decoder can be reduced to two reference frame memories and B-frames can be deleted as soon as they were displayed. The H.264 video standard allows arbitrary selection of reference frames from up to five reference pictures in the past or future.

In the case of bi-directional temporal prediction, the encoder must reorder the frames prior to the encoding step and the future reference P-frame is processed and transmitted before the enclosed B-frames. Figure 2.4 b) and c) show a typical IPB video sequence before and after reordering. Compared with the IP-only encoder, the IPB-encoder needs additional frame memories for picture reordering, at least a second reference frame memory and a significantly more powerful motion estimator.

Table 2.2: Coding mode name conventions.

Coding mode	**Picture structure**	**Reference frames**
I-frame only	I I I I ….	none
IP	I P P P P … I P P P P….	one or multiple past ref. frames
IP1	I P P P P …	one past ref. frame
IP2	I P P P P …	two past ref. frames
IP3	I P P P P …	three past ref. frames
IPX	I P P P P …	X past reference frames
IPB	e.g. IBP or IBBP	past and future ref. frames
IBP	**I** *B* P *B* P *B* P *B* P *B* **I** *B* P	1 past + 1 future ref. frame
IBBP	**I** *B B* P *B B* P *B B* **I** *B B* P	1 past + 1 future ref. frame

The most common GOP-structure for IPB-streams is IBBP, with 12 to 15 frames in a GOP and a maximum P-frame distance of two B-frames. Increasing the IP-frame distance does not necessarily increase the coding efficiency, as the temporal correlation of the IP-frames decreases. For high-motion sequences the IP-prediction over several B-frames requires an extended spatial search range which increases the computational complexity. Furthermore, the stronger changes in light, perspective and background result in larger prediction error.

Besides these common GOP-structures, other coding schemes like **I**P**I**P**I**P or **I**BB**I**BB**I** can be used which may be advantageous under certain computational constraints. In the two mentioned cases, the maximum prediction length is very short, so that error propagation due to numerical inaccuracies becomes less critical. For high data-rate transmissions it is even possible to avoid picture-reconstruction via IDCT and use the original image as reference instead.

2.3.3 Basic video coding tools

In this chapter a brief summary of the basic coding tools and their effectiveness is presented for the typical video test sequence *foreman* based on H.263 and MPEG-4 video codecs. The rate distortion curves in figure 2.5 [32] indicate the image quality which can be achieved at a certain data rate with the selected algorithms. The definition of the quality metric PSNR is given in section 3.2. A more detailed

analysis on H.264 temporal prediction modes will be presented in section 3.2. The uncompressed data rate in 4:2:0 color format is 12 bpp (bits per pixel).

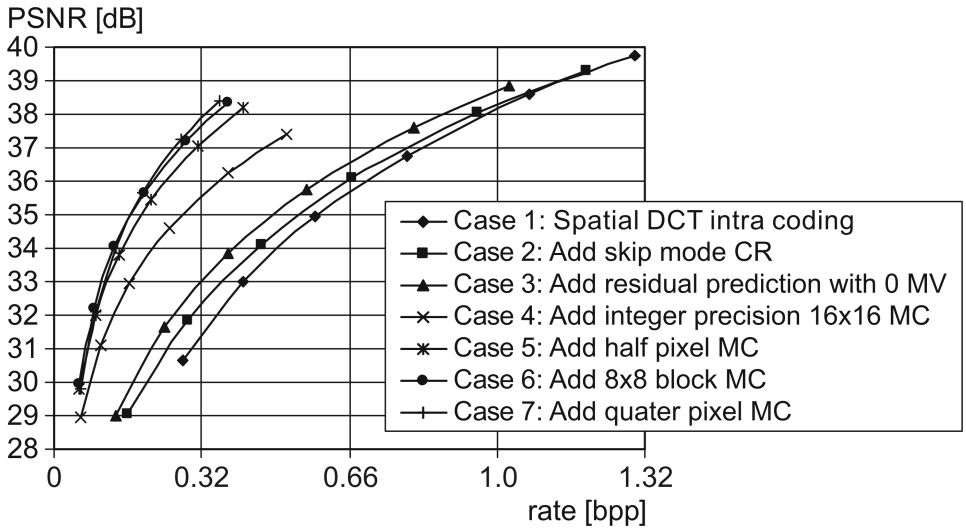


Figure 2.5: Effectiveness of basic coding tools ([32], test sequence *foreman*, CIF 382·288 pixels, 15 *fps*, 4:2:0 color format, H.263 / MPEG-4 part 2 video codecs).

Case 1 represents the basic intra coding method using the 8 by 8 DCT and entropy coding similar to the original JPEG codec. This coding mode can achieve excellent image quality at high bit rates of about 1.3 bits per pixel (bpp including luminance and chrominance channels).

The conditional replenishment method (CR, used in the H.120 compression standard) shown in *Case 2* can improve the image quality by selectively updating only frame regions which have changed significantly. This algorithm is realized by correlating the previous and current frame on a block basis and selecting only blocks with low correlation for the intra update. The simulation results indicate that for low target bit rates about 20 % data rate can be saved, but the gain decreases towards higher quality requirements. As the *foreman* sequence is recorded with an unstable, hand-held camera, this algorithm performs significantly poorer than in video conferencing applications with fixed cameras.

Case 3 introduces residual coding without motion compensation (i.e. forcing zero motion vectors in the video encoder) to the video codec, resulting in significant improvements and relative coding gains of 10 % to 25 % compared with the intra coding method. Once again this method does not perform very well with the unstable *foreman* video sequence.

The most significant improvements can be achieved by combining motion compensation with the residual coding method as shown by the *Case 4* simulation. The integer precision motion compensation can reduce the data rate by 45 % to 66 % with a significantly higher coding gain for low data rates. This coding mode was first applied in the H.261 video codec.

Already in the MPEG-1 standard fractional sample motion compensation with ½ pixel precision was introduced (*Case 5*), yielding significantly better performance especially at high data rates. Interestingly, the coding gain compared with integer precision motion compensation (MC) decreases towards lower data rates which may be caused by the fact that originally sharp contours are blurred and altered by coding artifacts if coarse quantization is used. In the MPEG-4 ASP (advanced simple profile) and H.264 standard, the motion vector precision was further increased to ¼ pixel precision which again increases the quality especially at high bit-rate applications (*Case 7*).

Video standards up to and including MPEG-2 and H.263 use 16 by 16 pixel macroblocks as basic unit for progressive video motion compensation. As indicated by the simulation *Case 6*, the coding performance can be further improved if macroblocks can be split into smaller units, each associated with a separate (differential) motion vector. The variable block size motion compensation (also known as macroblock partitioning) can be applied to regions of the basic transform size (4 by 4 pixels for H.264, 8 by 8 for all other standards) and shows significant gains at discontinuities along object borders.

In section 3.2 the performance of temporal prediction modes using multiple past reference frames and bi-directional prediction are evaluated in detail.

2.3.4 Temporal prediction tools

Implementing the motion estimation (ME) for temporal prediction along with the coding mode decision is one of the most critical decisions in the design of a video encoder. It is generally desirable to minimize the computational complexity of the motion estimation while keeping the loss of image quality at a minimum level. In the context of newer video standards like H.264, the motion estimation problem is strongly related to finding the optimal partitioning mode for the macroblock which increases the complexity of the mode decision algorithms and requires joint optimization of the motion estimation and mode selection.

The ME algorithms are generally not defined in the standards, and a variety of algorithms were developed for different applications. In the context of video coding, block matching algorithms are most commonly used which calculate the correlation of a block in the current frame and a region of the same size in a reference frame. It is generally assumed that the reference region with the highest correlation minimizes the energy in the error signal as well as the data rate for the residual coding. The basic operation sequence of modern motion estimation algorithms is summarized in table 2.3 and explained in the following sections.

Table 2.3: Motion estimation tools.

1. Motion vector prediction	Spatial or temporal MV prediction
2. Macroblock partitioning	Exhaustive methods Early termination methods
3. Motion search algorithm	Exhaustive search algorithms Fast search algorithms Hierarchical multi-resolution methods
4. Matching function	SSD, SAD, Integral Projections
5. Subpixel refinement	½ and ¼ pixel up-interpolation

Motion search algorithms

Considering the conceptual approaches of motion search algorithms, the following groups can be formed:

- Exhaustive motion search methods: all possible locations in a given search window of the reference frame are tested, i.e. full-search.
- Fast search algorithms typically employ gradient methods where an optimum search direction is iteratively determined by testing a set of sample locations, possibly searching a local minimum in this direction and continuing the search in the next iteration. Commonly used algorithms are 2D-log search, three step search (TSS) and diamond search.
- Hierarchical multi-resolution methods perform searches on different down-scaled image resolutions and project the best-matching motion vector to the next lower resolution image. As the computational cost for block matching in low-resolution images decreases with a power of two, larger search ranges can be covered in low-resolution images and the search window in the full-resolution image can be decreased. The gain in matching operations is however reduced by the complexity of down-scaling the image and storing it in fast-access memory.

A comparison in terms of quality and computational complexity of selected motion estimation algorithms is given in [29]. Concerning computational demands and implementation restrictions, the search range which defines the maximum distance of search postions from the zero location of a macroblock is a critical issue. Although modern video standards allow unrestricted motion vectors, it is

generally not necessary to consider the whole reference frame as potential reference for a certain macroblock. From the simulations presented in [33] it can be concluded that a search range of 100 by 70 pixels is sufficient for encoding SDTV video (720 by 480 pixels). This implies that a reference window covering about one third of the horizontal and vertical extension of the video frame is sufficient even for high-quality applications.

Motion vector prediction

A commonly used technique which can be combined with many search algorithms is motion vector prediction. As the motion vector field is usually continuous in large image regions, neighboring motion vectors are strongly correlated which can be exploited to reduce the computational complexity. If a good prediction for the current macroblock motion vector is available, the search range of exhaustive methods can be reduced and gradient methods converge faster to the optimal position. As the frame is coded from top to bottom and left to right, simple methods use the left neighbor and one or more top neighbors of the current macroblock for forming the prediction [21].

Matching functions

For measuring the similarity of image regions different methods can be applied. As the DCT is a energy-conserving transformation it is generally desirable to minimize the energy in the difference signal. Hence, the sum of squared differences (SSD) is one option for a similarity measure:

$$SSD = \sum_{i=0}^{N-1}\sum_{j=0}^{N-1}\big(a(i,j)-b(i,j)\big)^2, \tag{2.3}$$

where N is the size of the quadratic image region to be correlated and a and b are the pixels of the two frames to be compared. As the SSD computation requires computationally expensive multiplications, the simple sum of absolute differences (SAD) is widely used as it also delivers good results:

$$SAD = \sum_{i=0}^{N-1}\sum_{j=0}^{N-1}\big|a(i,j)-b(i,j)\big| \tag{2.4}$$

A third method which is rarely used although computationally less expensive in the combination with certain search algorithms is the integral projections method [34,35,85]:

$$PROJ = \sum_{j=0}^{N-1}\big|Ha(j)-Hb(j)\big| + \sum_{i=0}^{N-1}\big|Va(i)-Vb(i)\big| \tag{2.5}$$

$$Ha(j) = \sum_{i=0}^{N-1} a(i,j),\ Va(i) = \sum_{j=0}^{N-1} a(i,j) \tag{2.6}$$

Ha(j) is called a horizontal projection of the line *j* of pixels from the image data a, which is the sum of all *N* pixel values in this line. Similarly, the vertical projection

Va(i) is the sum of *N* pixel values of the i-th row of pixels. The similarity measure *PROJ* is calculated by subtracting the *N* horizontal projections of the current block and the reference block and accumulating the absolute values. Similarly the vertical projections are compared and a sum of the vertical and horizontal projection differences is formed. Although the equations look more complex, the integral projections matching can be implemented with fewer computations than the SAD method as the projections can be pre-calculated and re-used for any number of matching operations.

Macroblock partitioning

Because motion is not always constant for the complete macroblock, newer standards like H.264 introduced motion compensation for picture elements down to 8 by 8 or even 4 by 4 pixels. This can increase the coding efficiency but it is also computationally more expensive as a large variety of possible coding modes should be tested for each macroblock. The focus in the field of video encoding therefore lies in early termination strategies by applying iterative splitting of blocks in inhomogeneous frame regions [36]. An alternative decision criterion for block partitioning is presented in [37] targeting applications with true motion vector calculations like frame rate conversion. The criterion employs an SAD-based decision rule and an estimate for the expected SAD at certain displacement positions.

Subpixel refinement

For improving the coding efficiency motion vectors are calculated with half pixel precision (MPEG-1, MPEG-2) and quarter pixel precision (MPEG-4, H.264). Interpolating the pixels for the prediction reduces noise effects and decreases the error signal. The simulations in figure 2.5 as well as H.264 coding experiments [38] indicate that increasing the precision to $1/8$ pixel would be beneficial only in high-quality, high-resolution applications where detailed texture can be predicted more precisely with higher MV resolution. A theoretical analysis of optimal subpixel motion compensation selection is presented in [39]. It is also possible to derive a mathematical model for determining promising subpixel search positions by analyzing the local characteristics of the cost function [40].

2.3.5 Compressed video stream hierarchy

The concept of hierarchical stream syntax was first introduced in the H.261 video coding standard. In this section the hierarchy of the MPEG-2 video data stream is described but the concept is similarly implemented in all ITU and MPEG video standards.

The video standards generally define only the syntactic elements, the decoding process and restrictions on parameters. The encoder and implementation issues

are not part of the standards which allows further improvements even years after the standard was released as long as the resulting data stream conforms to the standard. Figure 2.6 shows the hierarchy of the syntactic stream elements [18,41]:

- Video sequence

  The video sequence is the highest level of the coded bit stream. It starts with a "sequence header" containing global parameters of the video sequence: image resolution, display aspect ratio, data rate, buffer sizes, profile and level information and user-defined quantization matrices. The sequence header must be present at the beginning of a video stream and may be repeated before any other picture in the stream. In many applications (i.e. DVD streams) the header is inserted at the beginning of every group of pictures to allow decoding start at $0.5\ s$ interval entry points. The end of a stream is marked by an "end of sequence" header which usually stops the decoding process.

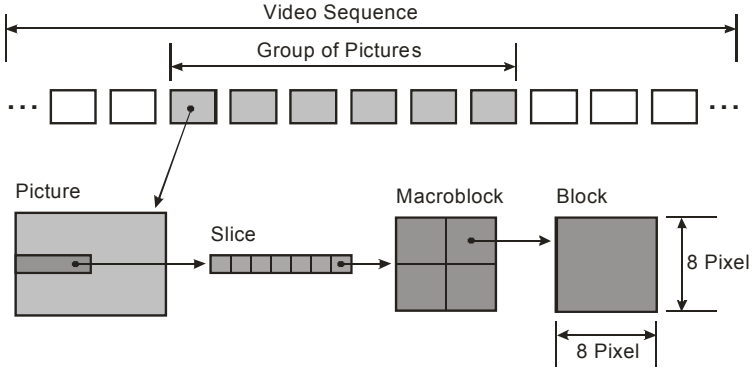


Figure 2.6: Hierarchy of the MPEG-2 data stream.

- **Group of pictures (GOP)**

  The term „group of pictures" was originally defined in the MPEG-1 standard as a sequence of coded images starting with an I-frame and an arbitrary number of P and B-frames. The term is not any more defined in newer standards but it usually refers to the MPEG-1 definition.

- **Picture**

  The "picture header" is obligatory in all standards for every image and contains information about frame or field coding, the temporal display order, and prediction type (I, P, B). In the case of MPEG-2 video sequences, additional information like field order, DC quantization precision and the range of the quantizer scale is included.

- **Slice**

  In the MPEG-1 and MPEG-2 standards, a slice is formed by a consecutive sequence of macroblocks which are all located in the same horizontal row of macroblocks. The slice header allows re-synchronization of the decoder if a transmission error occurred within a picture, therefore it contains all necessary data for decoding the next macroblock even if preceding macroblocks are missing. The H.264 standard allows more flexible slice structures for different target applications with increased error robustness or reduced syntax overhead for error-free media.

- **Macroblock**

  Each macroblock contains the luminance and chrominance information of a 16 by 16 pixel image region. For the standard 4:2:0 color sub-sampling mode, a macroblock is formed by four 8 by 8 pixel luminance sub-blocks and two 8 by 8 chrominance sub blocks. In MPEG-2, the motion compensation is performed on the macroblock-level and the macroblock header contains coding mode information (intra, predicted, skipped, interlaced) and up to four motion vectors which are coded differentially with respect to internal motion vector predictors. In H.264 the motion compensation can be performed on individual DCT blocks rather than macroblocks.

- **Block**

  In the MPEG-2 video stream, a block describes luminance or chrominance information of an 8 by 8 pixel image region. The two-dimensional DCT is performed on these sub blocks which results in one DC-coefficient representing the average brightness and 63 AC-coefficients representing texture information. The DC-coefficient is encoded differentially with respect to the last block at a fixed precision. AC-coefficients are quantized with different precisions followed by an entropy coding stage.

  In the H.264 standard, the transformation block size was reduced to 4 by 4 pixels. The H.264 stream syntax allows however flexible motion compensation on 16 by 16 down to 4 by 4 pixel sub block areas.

The largest amount of data in a medium or high quality MPEG-2 data stream is made up by DCT residual and motion vector information which is entropy encoded with variable bit-length using pre-defined Huffman tables. As the Huffman code does not offer any re-synchronization mechanism, a single bit-error leads to a non-recoverable error in the decoder and the following data cannot be decoded. To enable fast re-synchronization with a corrupted stream, the beginning of slices, pictures and video sequences are marked with unique hexadecimal “0x000001”

headers which are aligned to byte positions. As macroblocks don't offer any means of re-synchronization, a single bit error usually leads to a complete loss of all following macroblocks in the line up to the next slice header. Consequently bit errors within I or P-pictures which are used as reference frames cause severe display errors lasting for up to 0.5 seconds until the next I-frame is received.

Newer standards like MPEG-4 and H.264 allow more flexible ways of placing re-synchronization markers in the data stream allowing better adaptation of the coding to the transmission media. In the MPEG-4 standard re-synchronization markers are inserted periodically so that the bit stream is split into packets with equal numbers of bits rather than a constant number of macroblocks as in the MPEG-2 standard [42,43]. A superior performance can be achieved because the MPEG-2 slices have strongly varying data rates which results in a higher error probability of slices with significant texture and motion. Furthermore, the MPEG-4 standard introduced a data partitioning mode where all motion vector data of a group of macroblocks is transmitted before the texture information. If a transmission error occurs in the texture part of a packet, the motion vectors can still be used for forming the temporal prediction and concealing the error on the display.

In the H.264 standard the flexible macroblock ordering (FMO) coding tool is introduced for additional error protection [44]. The fundamental idea is that the visual effect of a corrupted macroblock can be concealed if it is surrounded by many correctly decoded macroblocks. This can be achieved by assigning the macroblocks of a frame to different slice groups which are transmitted sequentially in different packets. Neighboring macroblocks can be transmitted in different slice groups (e.g. checkerboard with two slice groups) and if the data of one slice group is corrupted, error concealment for the missing macroblocks can be applied using motion vectors or intra prediction from the surrounding macroblocks of other slice groups.

2.3.6 Digital video representation

2.3.6.1 Video sampling parameters

For digital processing, the images are sampled at fixed spatial resolutions in x and y direction with an orthogonal raster. Each sampling point is called a pixel consisting of three discrete values describing brightness and color information.

The definition of the common video sampling parameters originates in the ITU-T BT.601 standard for sampling analog PAL and NTSC video signals [41]. The standard definition television (SDTV) resolution is derived from an orthogonal, line-locked scanning raster with 13.5 *MHz* luminance sampling frequency. Smaller

image formats as listed in table 2.4 are derived by subsampling each component with a factor of 2.

Table 2.4: Standard video frame sizes and frame rates.

	"PAL" format	"NTSC" format
SDTV	720 · 576 *pixels* 25 *fps* / 50 *fps* interlaced	720 · 480 *pixels* 30 *fps* / 60 *fps* interlaced
CIF	360 · 288 *pixels* *25 fps*	360 · 240 *pixels* *30 fps*
QCIF	180 · 144 *pixels* 25 *fps*	180 · 120 *pixels* 30 *fps*

The down-conversion from SDTV to CIF format can be achieved by discarding every second field of interlaced video. Therefore, video formats below SDTV are usually progressive video sequences even if they originate from interlaced source material. The 13.5 *MHz* luminance sampling frequency results in slightly unbalanced x and y resolution (i.e. non square pixels) which leads to distorted picture presentation if the images are displayed on a square grid.

2.3.6.2 Frame and field pictures

In case of interlaced video, scanning is performed with double temporal resolution at half vertical image resolution. This results in two field pictures of two time instances with one field containing all odd lines (top field) and one field with the even lines (bottom field). Efficient coding of interlaced video was one of the major enhancements of the MPEG-2 standard. Figure 2.7 illustrates the different picture structures in the case of interlaced and progressive coding modes.

Field mode

Each field of the video sequence is encoded as a separate **field picture**, i.e. PAL video material is encoded with 50 field pictures per second with 720 by 288 pixels. The temporal prediction of odd fields from even fields and vice-versa usually involves a ½ pixel vertical shift.

Frame mode

A sampled video image is called a **progressive frame picture**, if all lines of the image are stored in one progressive frame and originate from the same time instance. Encoding progressive source material as frame pictures generally results in better compression than field picture coding as directly neighboring lines of video have higher correlation than second neighbor lines.

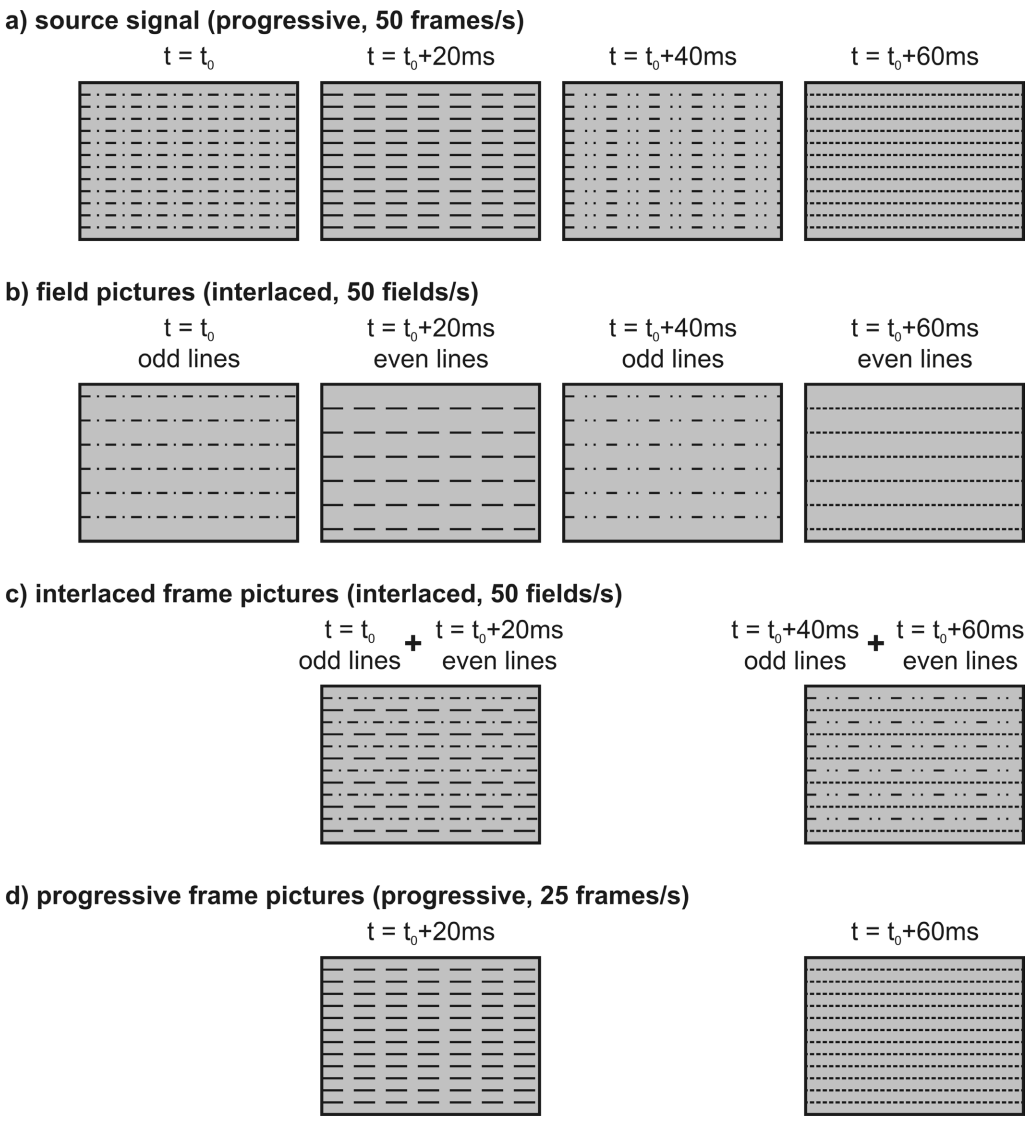


Figure 2.7: Frame and field picture definitions.

Encoding interlaced source material in frame mode can be realized by re-arranging ("weaving") the odd and even lines together into a single image before further processing is performed. The **interlaced frame picture** has correct spatial arrangement but as the two fields originate from different time instances they represent different phases of motion. Especially for fast-moving objects this results in vertical comb-like artifacts generating high-amplitude DCT coefficients with a significant loss in coding efficiency.

To overcome this problem, the lines within a macroblock can be adaptively rearranged before the DCT is performed. In the case of MPEG-2, each of the four luminance blocks of a macroblock then contains only information from one field which eliminates the comb artifacts. For predictive macroblocks different motion vectors for the two field sections can be transmitted, resulting in up to four motion vectors for a bi-directional predicted macroblock. In H.264 this coding tool is referred as macroblock-adaptive field-frame coding (MBAFF).

2.3.6.3 Color spaces

The representation of the pixel values in digital domain can be either in RGB or YCbCr color space [41]. In RGB color space three numbers represent the intensities of the red, green and blue color channel for each pixel. The range for each value is assumed a fractional number from 0.0 to 1.0 where 1.0 represents maximum intensity. Thus RGB=(0,0,0) representing black and RGB=(1,1,1) representing white. For eight bit standard quantization, 1.0 is mapped to the discrete value of 255.

A YCbCr value-triple consists of the luminance sample Y and two color-difference signals Cr and Cb which are linearly scaled versions of the YUV color space used in analog TV. The following conversion rules apply for conversion between gamma-corrected R'G'B' and YCbCr color space:

$$\begin{aligned} Y &= 0.299\ R' + 0.587\ G' + 0.114\ B' \\ Cb &= -0.127\ R' - 0.339\ G' + 0.511 B' + 128 \\ Cr &= 0.511\ R' - 0.428\ G' - 0.083\ B' + 128 \end{aligned} \tag{2.7}$$

For back-conversion:

$$\begin{aligned} R' &= Y + 1.371\ (Cr-128) \\ G' &= Y - 0.698\ (Cr-128) - 0.336\ (Cb-128) \\ B' &= Y + 1.732\ (Cb-128) \end{aligned} \tag{2.8}$$

2.3.6.4 Color subsampling

As the human visual system is not very sensitive to spatial resolution in the color components, color subsampling is frequently performed on the Cb and Cr components. The subsampling modes are identified as follows [41]:

4:4:4 Equal resolution for the Y, Cb and Cr components (no color subsampling).

4:2:2 The horizontal resolution of each color component is half the luminance resolution. No vertical subsampling.

4:2:0 The horizontal and vertical resolution of each color component is half the luminance resolution.

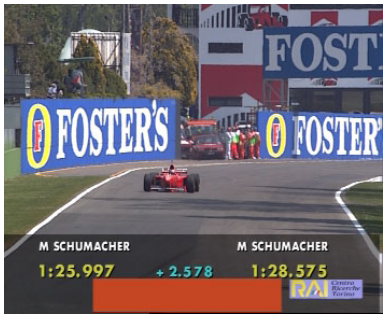
Conversion from the 4:4:4 to 4:2:2 format is generally uncritical, as no operations in the vertical direction are involved. In contrast to this, conversion algorithms for the 4:2:0 format must consider the picture structure, especially if fields are processed as interlaced frame pictures. For instance, applying a vertical conversion filter as for progressive frames on interlaced frame pictures results in temporal errors in the color components with striking effects in fast-moving video sequences.

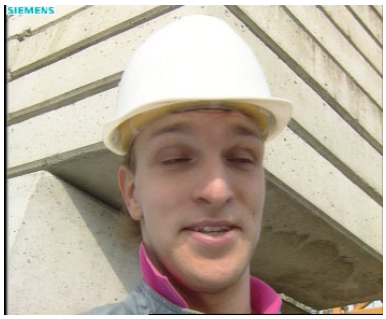
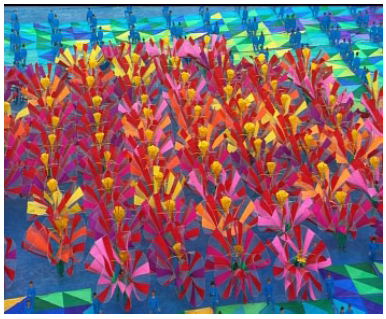
3 Video Coding Mode Analysis

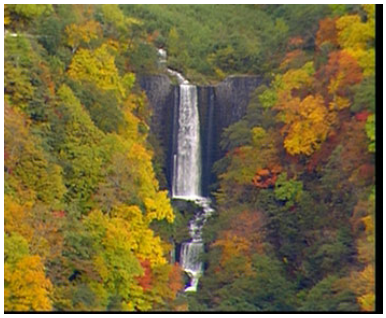
3.1 Test Sequences

For the simulations presented in this document a commonly used set of ten video sequences was utilized, covering a representative range of content from video conferencing to sports. The test sequences are sorted by decreasing motion activity, and the characteristics of each sequence are shortly described in table 3.1. The motion vectors for the first frames are also illustrated in figure 5.9 on page 105. All test sequences are progressive format with CIF resolution ($352 \cdot 288$ pixels) and 25 frames per second. Some sequences were converted from interlaced source material by discarding every second field and subsampling the horizontal resolution.

Table 3.1: Characteristics of test sequences.

	***canoe*** A high-motion sports video clip showing a camera pan with changing directions. The background is blurred by the motion (about 11 pixels per frame) and shows more texture when the camera movement slows down after two seconds. The water is flowing to the left side which is not detected correctly by the motion estimators due to the low amount of texture.
	***bus*** A high-motion camera pan to the left following the bus. Strong texture and uniform motion in the foreground and background allow precise motion estimation.
	***F1 car*** Video clip with varying camera pan and tilt. The amount of texture in the foreground road surface is fairly low compared with the background. As the camera is moving down, the amount of low-texture road surface increases.

	***foreman*** The hand-held camera results in variable motion of the complete image. In the first frames, the person is moving its head to the right.
	***flowergarden*** A moving camera video clip with uniform motion. The foreground has a high amount of texture and shows higher motion than the background. Due to the low amount of texture in the background, this video sequence has the largest local rate variations in intra coding mode.
	***mobile & calendar*** This video clip has complex slow motion in different directions. Due to the large amount of unique texture, the motion estimation works accurately. The amount of texture is quite uniform throughout the image.
	***barcelona*** A colorful video clip showing a camera zoom-out. The texture allows accurate motion estimation and the amount of texture is quite uniform throughout the image.
	***tempete*** A studio shot with very saturated colors showing a camera zoom-out with slight upwards movement.

	***paris*** A video conferencing scenario with still camera. The stable background has high contrast texture and only the persons are moving.
	***waterfall*** This video clip shows a high-texture forest with a slow camera zoom-out. The amount of texture in the image is quite uniform.

3.2 Definitions and quality metrics

Normalized data rate

Throughout this document the normalized data rate *r* counted in bits per pixel (bpp) is applied for comparison of data rates. It is calculated by dividing the file size *b* by the number of pixels represented by the compressed file of *N* frames of size *x* by *y* pixels.

$$r = \frac{b}{x \bullet y \bullet N} \tag{3.1}$$

The normalized data rate can be rescaled easily to the absolute data rate *R* at different frame sizes and frame rates, e.g. for CIF resolution test sequences the following conversion applies

$$R = \frac{b}{t} = \frac{r}{\text{bit}/\text{pixel}} \bullet \text{bit}/\text{pixel} \left(352 \bullet 288 \frac{\text{pixel}}{\text{frame}} \bullet 25 \frac{\text{frame}}{\text{s}} \right) = \frac{r}{\text{bit}/\text{pixel}} \bullet 2.534 \frac{\text{Mbit}}{\text{s}}. \tag{3.2}$$

Furthermore, the compression ratio can be calculated with respect to the uncompressed YCbCr input stream at eight bit precision for 4:2:0 color subsampling as follows:

$$c_{4:2:0} = \frac{8 \cdot 1.5 \text{ bit}/\text{pixel}}{r} = \frac{12 \text{ bit}/\text{pixel}}{r}. \tag{3.3}$$

Image quality measures

In this document the peak signal to noise ratio (PSNR) is used for measuring the image quality. It is defined by the mean square error (MSE) of a video sequence of

N frames with x by y pixels resolution. A and B represent the pixel brightness levels of the original and decoded frames, n is the number of bits used for sampling the video signal.

$$MSE = \frac{1}{x \bullet y \bullet N} \sum_{i=1}^{N} \sum_{j=1}^{y} \sum_{k=1}^{x} \left[A_{i,j,k} - B_{i,j,k} \right] \tag{3.4}$$

$$PSNR = 10 \cdot \log_{10} \left(\frac{(2^n - 1)^2}{MSE} \right) \tag{3.5}$$

For the calculation of PSNR values over a sequence of frames, the PSNR values of single frames must be converted to non-logarithmic MSE values prior to the averaging. In this document, the PSNR is generally measured only for the luminance (Y) channel of the video sequence, as the color component channels do not deliver meaningful results. Besides the higher sensitivity of the HVS to coding artifacts in the luminance channel, the video codecs usually deliver quite similar visual quality in the chrominance channels when the same quantization parameters are used.

Although the PSNR quality measure is mathematically simple to implement it has some drawbacks which must be considered when the results are interpreted. On the one hand it should be noted that the PSNR cannot be used for comparing the visual quality of different video sequences. This can be seen in figure 3.1 where ten video sequences were encoded with the same quantizer settings which results in quality variations of roughly 4 dB. The video sequences with low amounts of texture generally appear to have a higher PSNR value than high texture video sequences. On the other hand the PSNR can be used as an effective measure for the quality optimization of a single video sequence if the characteristics of coding artifacts remain the same.

In recent times extensive research has been done in the field of automated video quality analysis, but these methods are not yet widely used by other researchers. The structural similarity measure (SSIM) [45,46] is a promising candidate for future research with better validity for different kinds of coding artifacts produced by different compression standards.

3.3 Comparative coding mode analysis

As discussed in section 2.3.2, the video codecs allow different temporal prediction modes which have a major impact on the quality of the compressed data stream. In this section, the MPEG-2 and H.264 video compression standards are evaluated. By analyzing the behavior of a representative set of video sequences with different coding modes, important characteristics like image quality, effectiveness of temporal prediction modes and absolute data rates are determined. Furthermore, the simulation results form the basis for the latency analysis in chapter 4.

A related analysis of statistical properties of MPEG-1 encoded video sequences was presented in [47]. The referenced work targeted rate modeling of rate variations on ATM (asynchronous transfer mode) networks caused by VBR operation of the video encoder.

3.3.1 Test set-up and quality adjustment

In the following sections a comparative analysis of different predictive coding modes of the MPEG-2 and H.264 standard is presented. At the end of this section a quality comparison of the two video coding standards will be given, but the major focus of this study lies in the derivation of characteristics which affect the latencies of compressed video transmission systems. Therefore, only three distinct operating points were examined at low ($30\ dB$), medium ($35\ dB$) and high ($40\ dB$) Y-PSNR quality. All tests were performed with constant quantizer value (without rate control), resulting in constant quality encoder operation with variable output data rate. In this way, the algorithmic performance of the video standards can be evaluated without measuring the effects of different rate control algorithms.

The problem of cross-standard video quality evaluation is non-trivial because different coding standards and frame types have different characteristics. A common way to approach this problem is to measure the rate-distortion curves for individual sequences at different coding modes with many simulation points. In this survey a different approach was taken to reduce the number of simulations.

The basic approach for the quality adjustment in the $35\ dB$ operating point is shown in figure 3.1, figure 3.2 and table 3.2. In the first step, the average video quality over the ten video sequences of section 3.1 (each 72 frames, approximately three seconds of video) was measured for the H.264 intra coding mode. The measured mean PSNR value (e.g. $34.84\ dB$) served as reference value for the other simulations at the same operating point.

In a second step, the quantizer value for predicted macroblocks in the IP and IPB coding modes were adjusted to match the average PSNR values of the individual intra-coded sequences (figure 3.1, table 3.2) as closely as possible. In this case, an individual adjustment for each video sequence was used, which allows precise analysis of the performance of temporal prediction tools.

In a third step the MPEG-2 encoder quantization for the intra mode was adjusted to the same average PSNR quality as the H.264 intra mode. Instead of adjusting all video sequences individually, a common quantizer value was selected which resulted in the best match of the average PSNR value over all sequences (e.g. $34.81\ dB$). The individual PSNR measures differ however significantly for some sequences as the H.264 and MPEG-2 transformation and quantization have different characteristics. Finally, the quantizer value for the predicted frames was

adjusted individually to a best match. The same procedure was applied for the $30\,dB$ and $40\,dB$ operating points with exact parameters and measures documented in the appendix table A.1 and table A.4.

Table 3.2: Quantization parameter adjustment for $35\,dB$ operating point.

	H.264		MPEG-2	
	QP_{intra}	QP_{pred}	QP_{intra}	QP_{pred}
canoe	31	29	14	15
bus	31	30	14	16
F1 car	31	29	14	16
foreman	31	30	14	15
flowergarden	31	29	14	18
mobile	31	29	14	17
barcelona	31	29	14	17
tempete	31	29	14	16
paris	31	29	14	17
waterfall	31	30	14	15
avg. PSNR	**34.84 *dB***	**34.79 *dB***	**34.81 *dB***	**34.80 *dB***

Because of the limited resolution of quantizer values, it is not possible to match the quality of intra and predicted coding modes exactly. In the chosen operating points, a minimum step of the quantizer results in roughly $0.3\,dB$ PSNR change for the H.264 video standard, respectively $0.1\,dB$ for the MPEG-2 standard. For the goal of this evaluation it is however sufficient to gain representative values of average and extreme performance of video encoders. Therefore, the evaluation of individual sequences does not consider the effect of small quality variations, as this would require at least three times more simulations for calculating linear approximations at the operating points. However, in section 3.3.5, the quality of the various coding modes based on 300 simulations is evaluated considering the rate and quality variations caused by the finite quantizer value resolution. The results indicate that the deviations of quality corrected simulations differ only in the range of a few percent from the uncorrected values. Furthermore, the minor quality differences do not affect the overall conclusions drawn from this evaluation.

The exact coding parameters for MPEG-2 and H.264 are documented in appendix F. The search range for motion estimation was set to +/- 16 pixels for both standards and all interlaced coding modes were inactive. For H.264 the constrained intra prediction was used which limits the intra prediction source pixels in P-frames to intra predicted macroblocks.

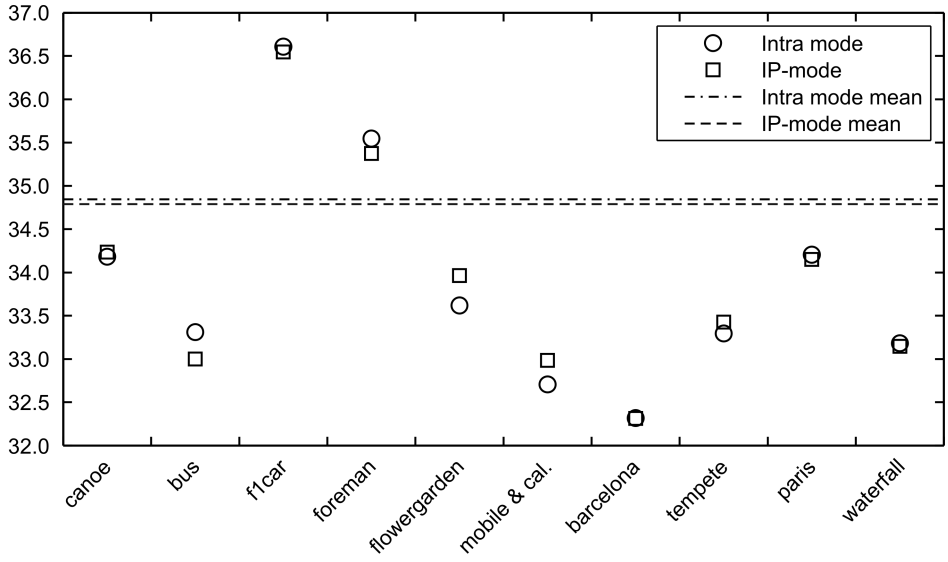


Figure 3.1: H.264 sequence PSNR at 35 *dB* operating point (appendix table A.2).

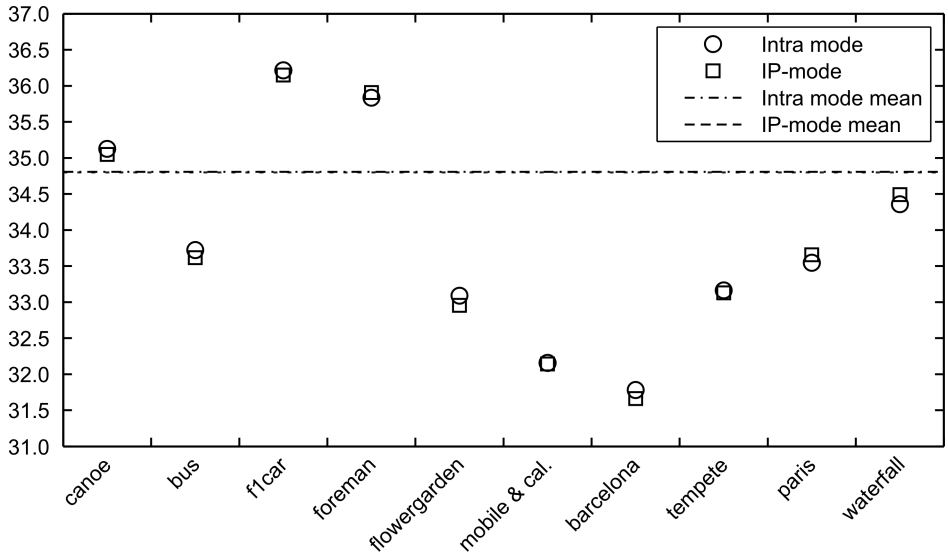


Figure 3.2: MPEG-2 sequence PSNR at 35 *dB* operating point (appendix table A.3).

3.3.2 Intra coding mode analysis

The minimum complexity intra coding mode was evaluated first and serves as a reference for the other simulations. Figure 3.3 shows representative results of the frame size measurement at a fixed quantizer value. The amount of data generated by a frame in the intra coding mode is mainly determined by the average amount of texture and the contrast of texture elements. Therefore, video sequences with high texture, e.g. *mobile & calendar* and *barcelona*, require more data rate than the low-texture sequence *foreman*.

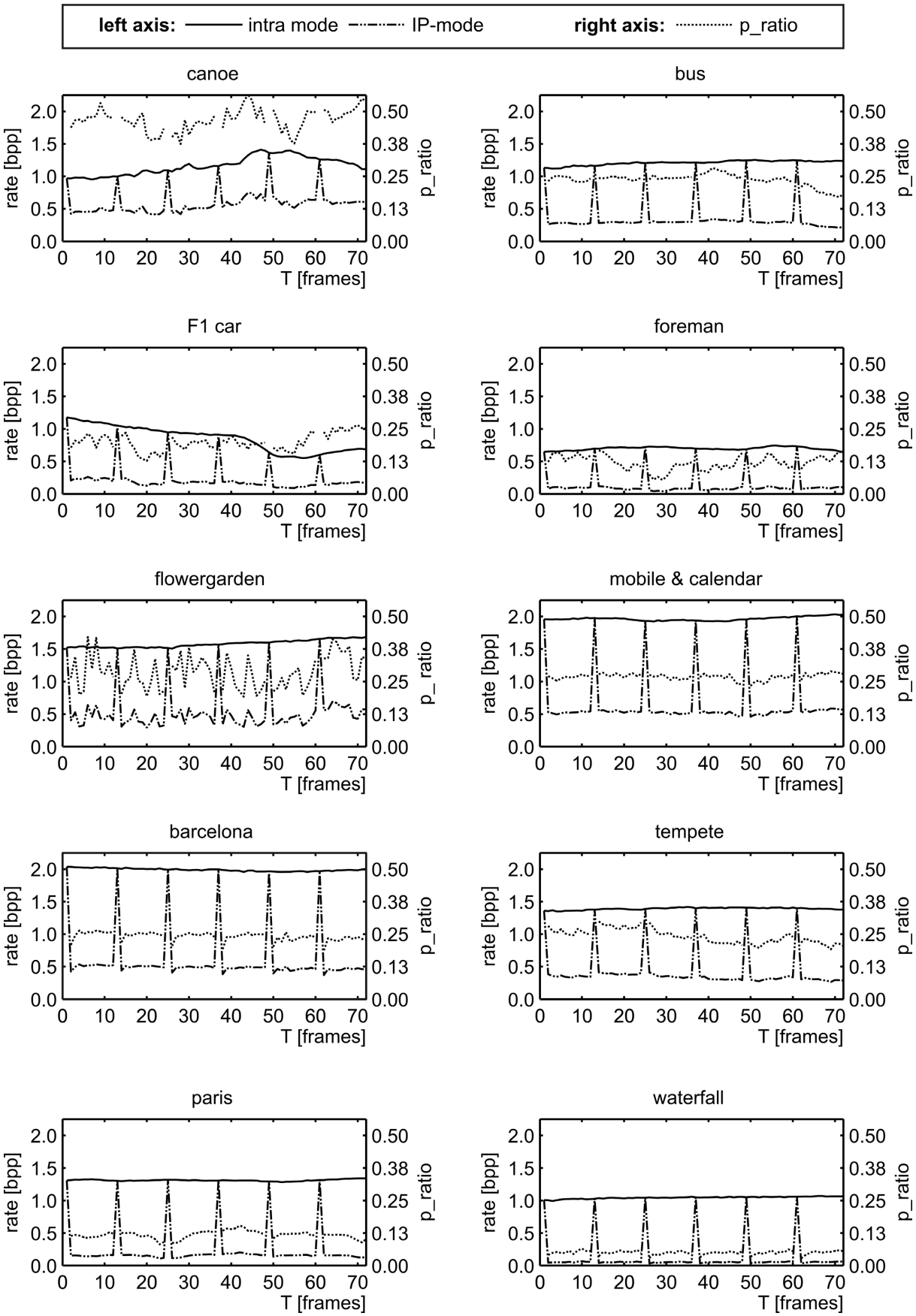


Figure 3.3: H.264 average bit rate per frame simulation results ($PSNR \approx 35\ dB$). The frame size ratio *p_ratio* is defined in Eq. 3.6.

In the intra coding mode, most of the video clips show only small variations over the 72 test frames. However, in the *canoe* clip the data rate increases when the camera motion slows down between frame 40 and 60, which brings up more texture in the background. Similarly, the *F1 car* sequence shows mainly low-texture road surface after frame 40 which reduces the data rate.

The frame-level simulation results for H.264 intra coding mode of all three operating points are summarized in figure 3.4. It can be observed that the mean data rate of the 30 *dB* and 35 *dB* operating points roughly scale by a factor of two whereas the 35 *dB* to 40 *dB* mean data rate scales by 1.75. Throughout all operating points, the *foreman* sequence has the lowest data rate but the maximum data rate shifts from *mobile & calendar* to *barcelona* towards higher quality. The *F1 car* and *canoe* sequences have the highest variance because of significant changes in texture due to camera movement.

The MPEG-2 simulations in figure 3.5 show slightly higher average data rates than H.264 but they are generally quite similar for the intra coding mode. For both video standards, the sequences *mobile & calendar* and *barcelona* require the highest data rates approximately 50 % above the average. The low-complexity *foreman* sequence is about 45 % below the average rate which results in rate variations of a factor three for different video sequences at the same quantizer values.

Comparing the simulation results in figure 3.4 and figure 3.5 indicates that the older MPEG-2 standard requires a data rate only six to eight percent higher than the H.264 standard to achieve the same average PSNR quality. This simulation does however not consider effects of the H.264 in-loop deblocking filter which improves the image quality especially for low bitrates.

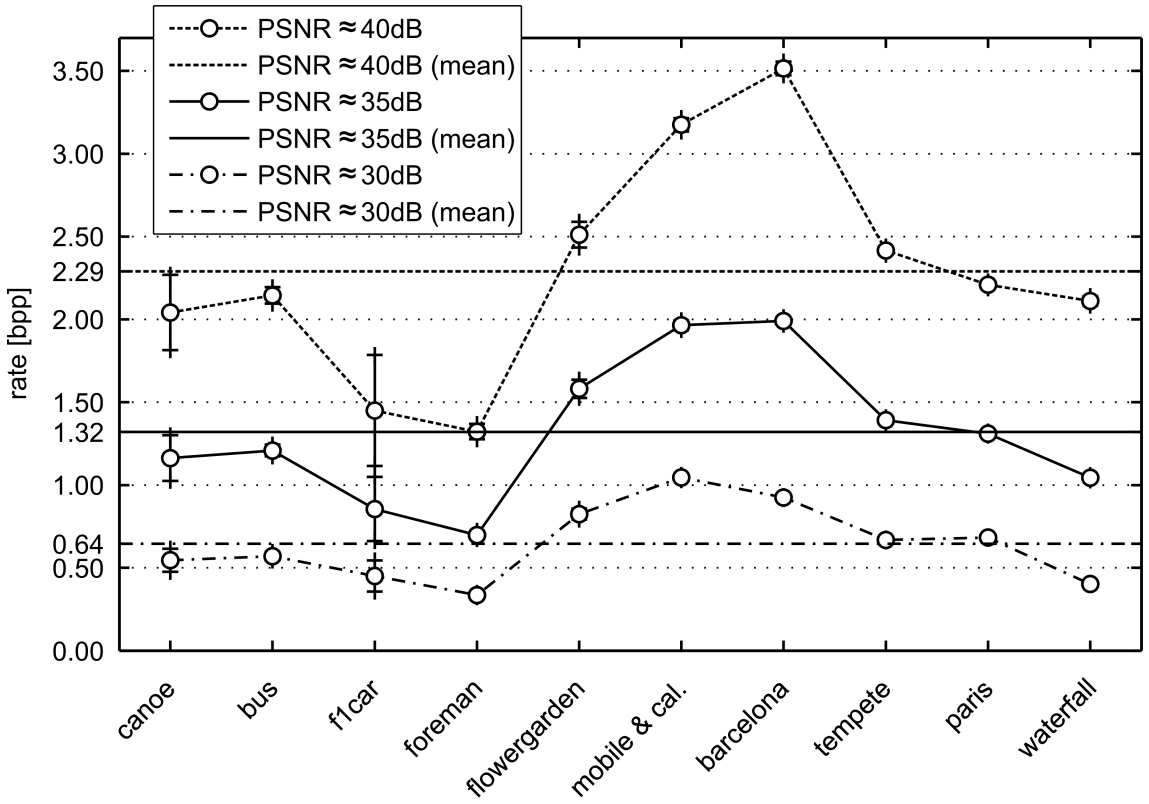


Figure 3.4: H.264 intra coding mode sequence data rates. Error bars indicate the variance of data rate.

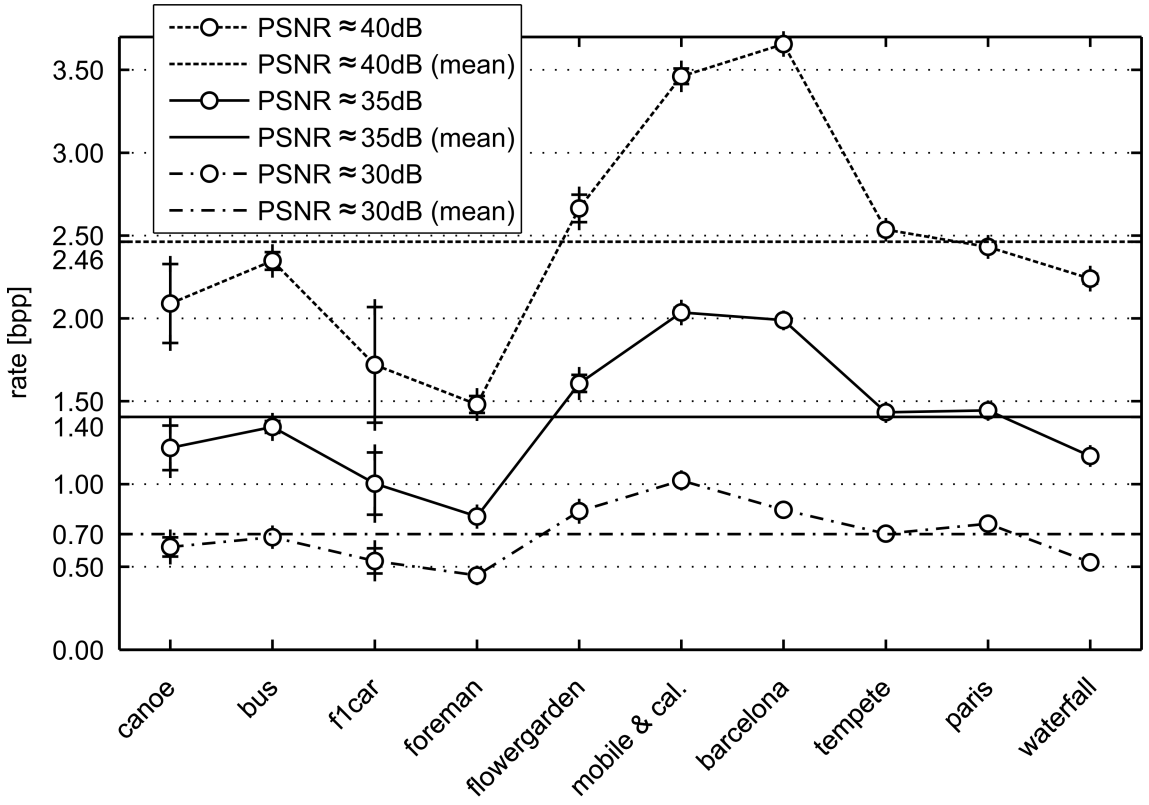


Figure 3.5: MPEG-2 intra coding mode sequence data rates. Error bars indicate the variance of data rate.

3.3.3 IP-coding mode analysis

Single reference frame forward prediction mode

In this section the performance of the forward temporal prediction mode with one past reference frame is evaluated. The GOP structure for all simulations is *IPPP..* with an I-frame period of 12 frames ($N_{GOP} = 12$). For the predicted frames, the quantizer value was adjusted individually for each sequence to match the average *PSNR* of the intra coding mode.

The data rate over T_{frame} plots are shown in figure 3.3 (page 39) for the H.264 video standard. On the right axis the P/I frame size ratio is plotted which is calculated by dividing the size of the P-frame by the size of the corresponding I-frame. Analyzing the plots it can be concluded, that the intra coding mode pictures consume about two to ten times more data rate than the predicted frames. The large intra-coded frames in the IP coding mode can be easily identified by the peaks in the frame size plots.

In the constant quality mode, the test sequence *canoe* shows long-term data rate variations which are caused by changing motion and texture. The intra to P-frame ratio stays however relatively constant throughout the video sequence. The *bus* sequence on the other hand shows only small rate variations and a higher P-frame gain. This is quite surprising, as the sequence has fast and complex motion combined with significant texture. The *foreman* sequence has a low-texture background which results in low data rates. However, the hand-held camera is very unstable which leads to significant rate variations over time in the predicted frames. The *flowergarden* sequence also shows significant rate variations in the p-frames which may be caused by unsteady operation of the motion estimation and mode decision due to the ambiguous texture of the scene.

Figure 3.6 summarizes the average frame-level data rate measures for the intra and IP coding modes of the simulations in figure 3.3. For the $35\ dB$ operating point, the predictive coding mode reduces the absolute data rate by $70\ \%$. The frame-level data rate for the predictive coding mode varies by a factor of five compared with a factor of three for the intra coding mode. For a constant bitrate transmission, the predictive coding mode therefore results in stronger quality variations than the intra coding mode.

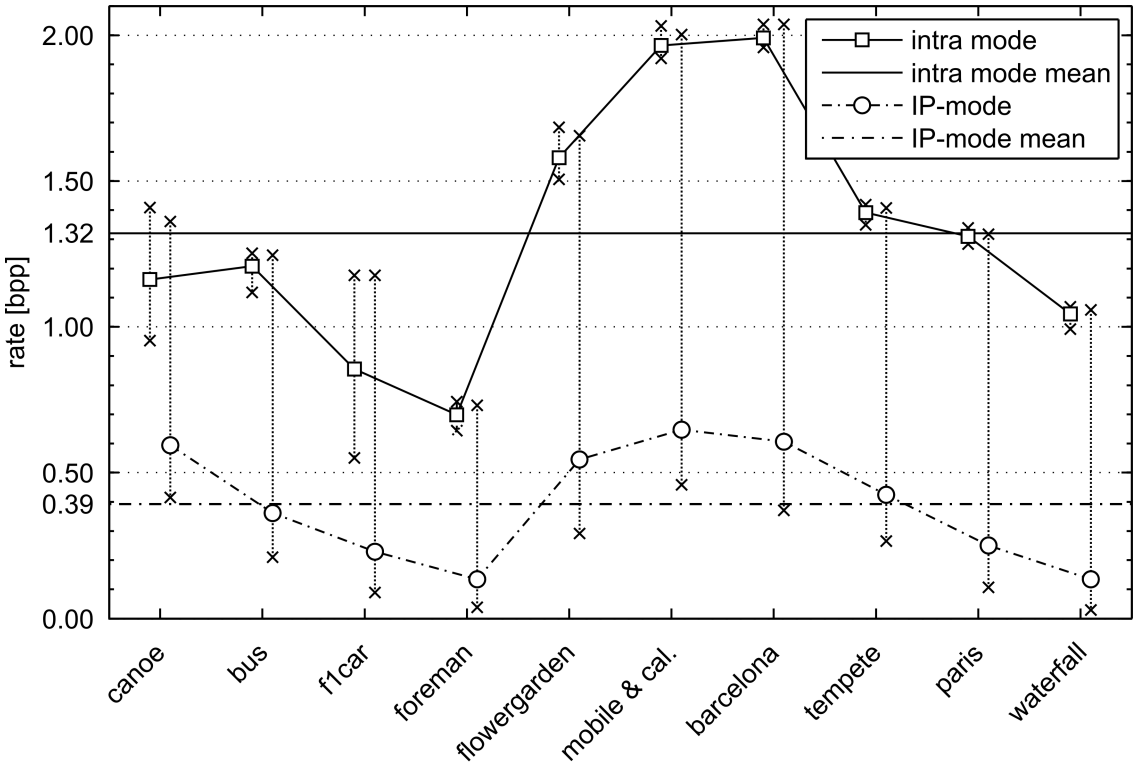


Figure 3.6: H.264 intra and IP mode bit rate measures ($PSNR \approx 35\ dB$, $N_{GOP} = 12$). The extended dotted error bars indicate the bit rate of the largest and smallest frame of the video sequence.

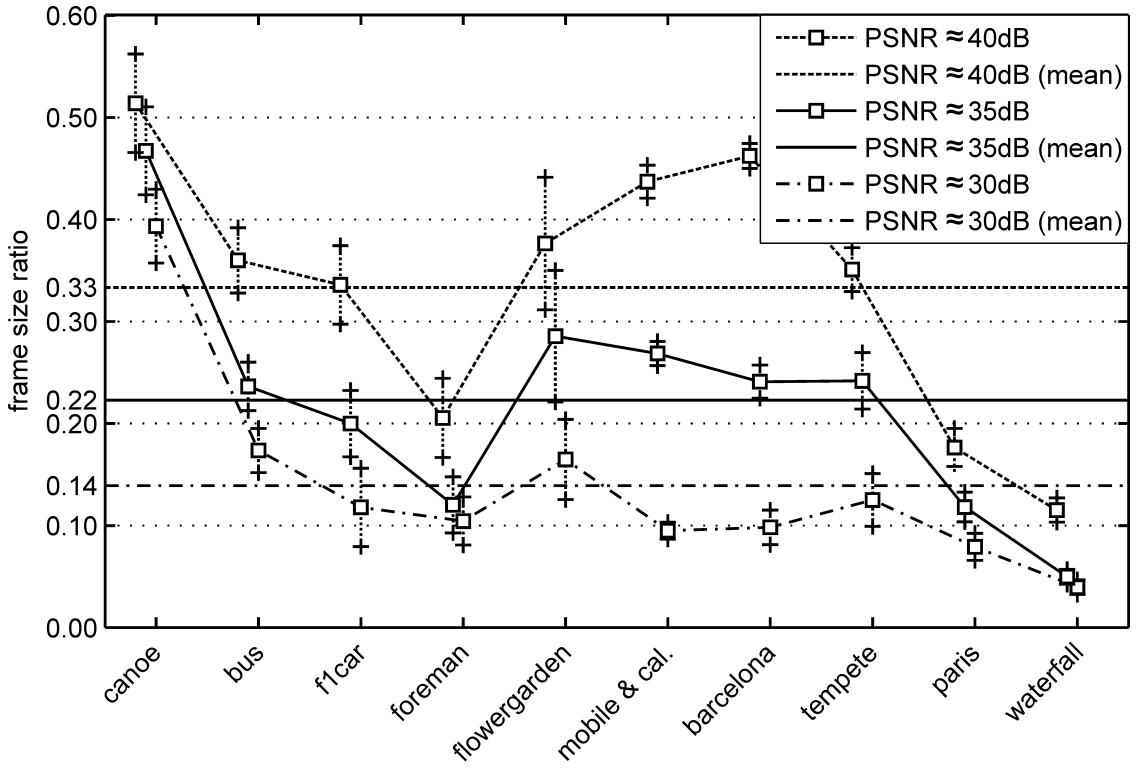


Figure 3.7: H.264 *p_ratio* calculated by the average size ratio of predicted frames and the corresponding intra-coded frames.

The effectiveness of the temporal prediction is evaluated by calculating the ratio of the size of each P-frame with its corresponding I-frame in the intra-coded video sequence. Frame-level plots are presented in figure 3.3 (right axis), with the I-frames being ignored. From the frame-level data, the average frame size ratio is calculated by

$$p_ratio = \frac{\sum_i \text{size of P-frame } i \text{ in IP-mode}}{\sum_i \text{size of P-frame } i \text{ in intra mode}}, \quad \begin{array}{l}\text{for all P-frames with index } i \\ \text{of IP-coded video sequence.}\end{array} \tag{3.6}$$

A good temporal prediction results in a small *p_ratio* and a high P-frame coding gain. Similar to equation 3.6, the average frame size ratio for B-frames is defined as:

$$b_ratio = \frac{\sum_i \text{size of B-frame } i \text{ in IPB-mode}}{\sum_i \text{size of B-frame } i \text{ in intra mode}}, \quad \begin{array}{l}\text{for all B-frames with index } i \\ \text{of IPB-coded video sequence.}\end{array} \tag{3.7}$$

Figure 3.7 summarizes the *p_ratio* for the H.264 IP coding mode for the selected test sequences at all operating points. The analysis shows that there exist strong variations in the P-frame coding gain with a wide range of 95 % down to only 50 % rate savings for predicted frames. Considering the ten video sequences at three operating points it can be clearly seen that the *p_ratio* decreases towards lower quality operating points. Hence, the temporal prediction is more effective for low-bitrate applications than for high-bitrate applications where the encoder may choose to transmit intra-coded macroblocks rather than temporal prediction residuals.

For the 30 *dB* operating point the coding gain of predicted frames mostly increases with decreasing motion activity. This is however not valid for the medium and high quality operating points where test sequences with strong texture and medium motion activity show higher *p_ratio* than high-motion sequences. For all coding modes, the *canoe* sequence has the lowest P-frame coding gain whereas the *waterfall* sequence has the highest P-frame coding gain.[1]

[1] The *canoe* sequence shows a bad temporal coding gain with a frame size ratio of 0.4 to 0.5 for the H.264 standard and even worse performance for the MPEG-2 standard. An in-depth analysis of the *canoe* stream reveals that the chaotic motion of the water in the lower image section is not tracked correctly by the full-search motion estimator. It was verified that increasing the motion search range does not lead to significant improvements in all cases. Presumably, the small amount of texture in the water hinders the motion estimation but it also results in low intra-coding costs. Hence, both reference encoders prefer to intra-code large sections of the lower image region rather than coding large differential motion vectors caused by bad motion estimation.

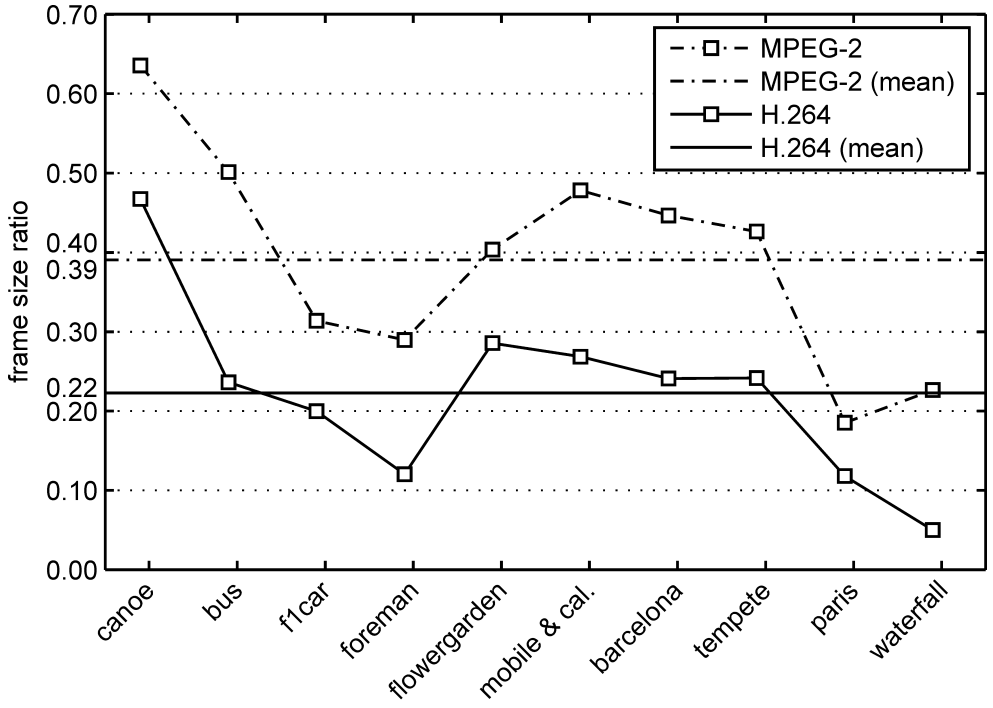


Figure 3.8: *p_ratio* for H.264 and MPEG-2 ($PSNR \approx 35\ dB$).

Figure 3.8 compares the frame size ratios for both video standards. It can be seen that the H.264 video encoder generally achieves a significantly better *p_ratio* than the MPEG-2 encoder. Taking into account the effect that the H.264 encoder only achieves about $10\ \%$ improved performance in the intra coding mode, it can be concluded that the major improvements of the H.264 standard come from the enhanced temporal prediction tools.

For the MPEG-2 video encoder, the average frame size of the P-frames is $22\ \%$ of the intra frame size. Compared with the newer H.264 video coding standard, MPEG-2 achieves only half the coding gain for P-frames. The outlier *canoe* is only 1.6 times above the mean *p_ratio* compared to 2.1 for the H.264 standard. The variations in the P-frame coding gain generally seem to be smaller in MPEG-2 than in H.264.

Enhanced forward prediction modes

Both, the H.264 as well as the MPEG-2 video standard offer enhanced temporal prediction modes with more than one past reference frame. The MPEG-2 "dual prime" prediction mode is however limited to interlaced P-frames and cannot be used with the progressive test sequences.

The H.264 video encoder can use up to five reference frames for temporal prediction, which may be all past or some past and future frames of the video sequence. The evaluation for one to five past reference frames was carried out for the $35\ dB$ operating point and the results are shown in figure 3.9 and summarized in table 2.1.

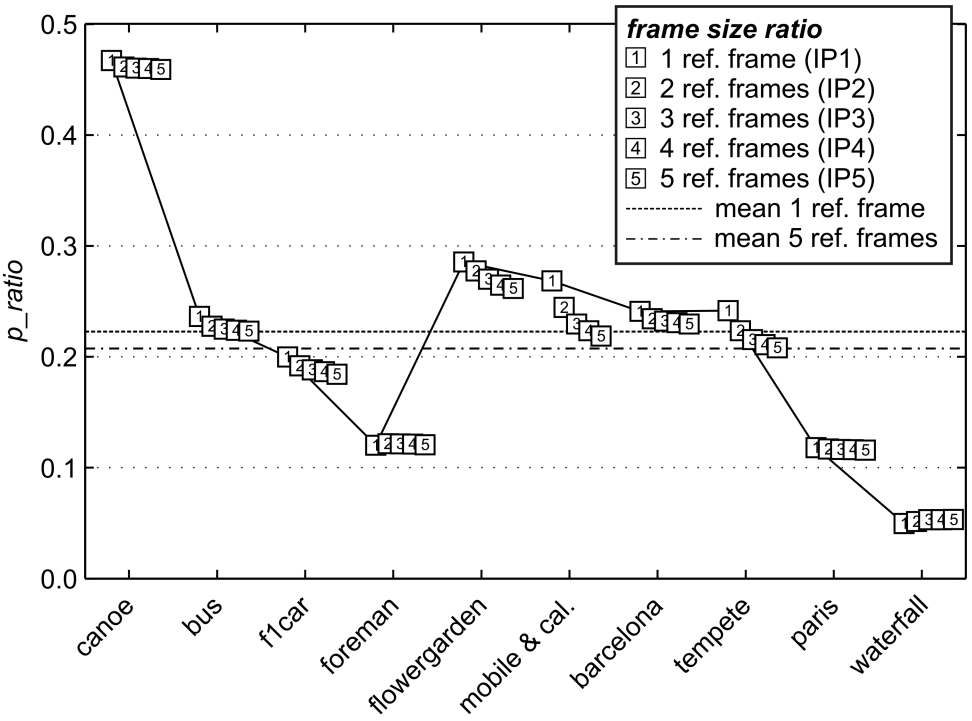


Figure 3.9: H.264 forward temporal prediction with multiple past reference frames ($PSNR \approx 35\ dB$).

The average performance gain is 3.59 % for two reference frames and 7.17 % percent for five reference frames compared with the simple forward prediction mode. Analyzing the individual results of the video sequences it can be seen that the three sequences *foreman*, *paris* and *waterfall* with very high p-frame compression do not profit from the additional reference frames. On the other hand, the high texture video sequence *mobile & calendar* as well as the *tempete* have significant gains of 18.5 % and 13.9 % respectively. The other five sequences show only minor gains in the range of 5 percent.

Table 3.3: Coding gains of H.264 multiple reference frame forward temporal prediction ($PSNR \approx 35\ dB$).

Reference frames	**Mean** *p_ratio*	**Gain**
1 (IP1)	0.223	0.00 %
2 (IP2)	0.215	3.59 %
3 (IP3)	0.211	5,38 %
4 (IP4)	0.209	6.27 %
5 (IP5)	0.207	7.17 %

It can be concluded that the H.264 enhanced forward temporal prediction can give a significant data rate reduction only in some special cases at the cost of significantly increased computation power for the additional motion estimation. As the correlation of frames decreases quickly with the temporal distance, increasing the number of reference frames to more than three frames does not seem reasonable.

3.3.4 IPB coding mode analysis

The simulations with the bi-directional temporal prediction modes were performed with a P-frame period of two (IBP) and three frames (IBBP) as defined in section 2.3.4.

Similar to the IP coding mode, the distance of the I-frames is defined by N_{GOP}. For the H.264 simulations, the same quantizer value was used for the P and B-frames, which results in an sufficiently equal quality throughout the video sequences (see table A.2 in the appendix). In the case of the MPEG-2 simulations, a manual adjustment of the B-frames quantizer value was necessary to level out the quality (table A.3). In any case, the quality of the B-frames depends significantly on the quality of the surrounding P-frames which has a major impact on the complexity of a rate control algorithm. The coding experiments show that it is generally not a valid assumption that equal quantizer values result in the same $PSNR$ quality for different frame types.

In figure 3.10, the frame size ratios for H.264 bi-directional prediction modes are compared with the simple forward IP prediction mode performance. For the bi-directional prediction modes, the B-frames achieve a $33\,\%$ to $82\,\%$ better compression ratio than the surrounding P-frames. The significant B-frame coding gain is however often cancelled out by the degraded P-frame coding performance caused by the temporal prediction across the enclosed B-frames. Hence, the overall performance is only about eight percent better compared with the IP-coding mode.

Analyzing the individual sequences it can be seen that the bi-directional coding modes have significant gains with some sequences (i.e. *flowergarden*, *mobile & calendar*, *barcelona*, *tempete*), whereas all other sequences show only insignificant improvements. Three out of these four sequences have also significant gains with multi reference forward prediction modes (figure 3.9). The sequences with high bi-directional coding gain can be characterized by moderate motion activity and significant amounts of texture, as they also require the highest intra-coding mode data rates (figure 3.5). Due to the high motion activity, the *canoe* sequence even shows excess data rates for the IBBP coding mode since it requires motion estimation over three frames with an estimated vertical offset of 40 to 50 pixels.

Figure 3.11 shows the simulation results for MPEG-2 bi-directional prediction. The bi-directional coding gains generally look less significant, but the relative gain compared with the IP coding mode is equal to the H.264 simulation results. The B-frames achieve an average 45 % rate reduction compared with the surrounding P reference frames. The top-four sequences with the highest IPB coding gains are equal to the H.264 simulations, but three other sequences (i.e. *bus*, *foreman* and *waterfall*) also show significant gains. In the high-motion video sequence *canoe*, the forward temporal prediction for the P-frames fails completely in both cases as the vertical search range is too small. It was verified that increasing the search range does not improve the performance in this case. The selection of IBP or IBBP coding mode seems to be uncritical in terms of image quality.

Concerning the selection between an IBP or IBBP coding mode, the simulations indicate that both coding modes result in the same effective frame size ratios, but the IBP mode shows more stable behavior for high-motion video sequences. Based on these observations, a recommendation for using the IBP rather than the IBBP GOP structure can be given.

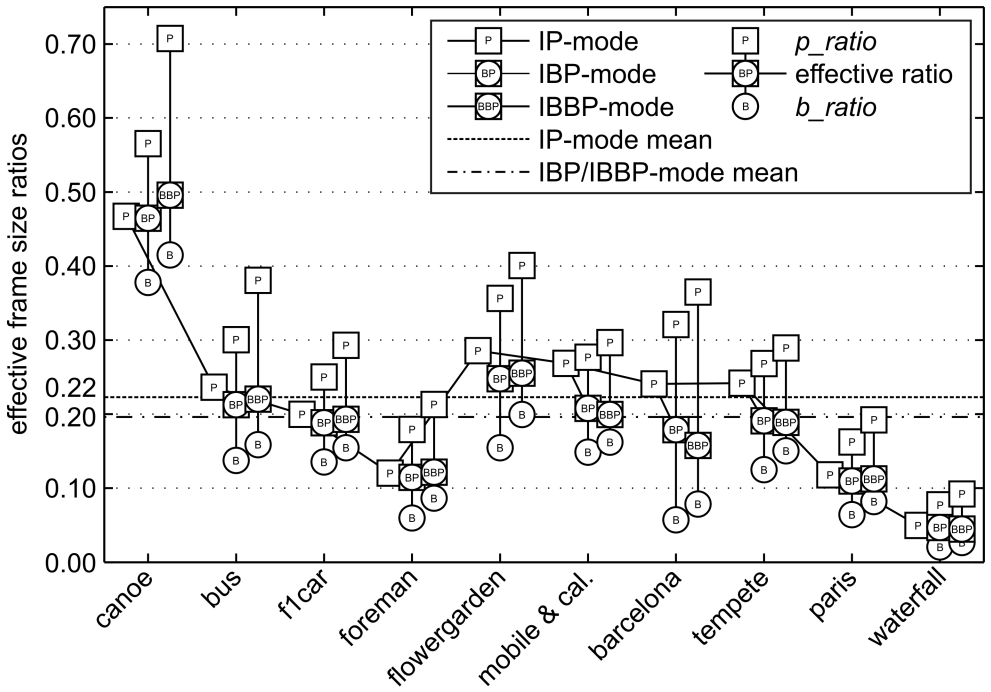


Figure 3.10: H.264 bi-directional prediction frame size ratios ($N_{GOP} = 12$, $PSNR \approx 35\ dB$; one future, one past reference frame; IBP and IBBP mean values are identical).

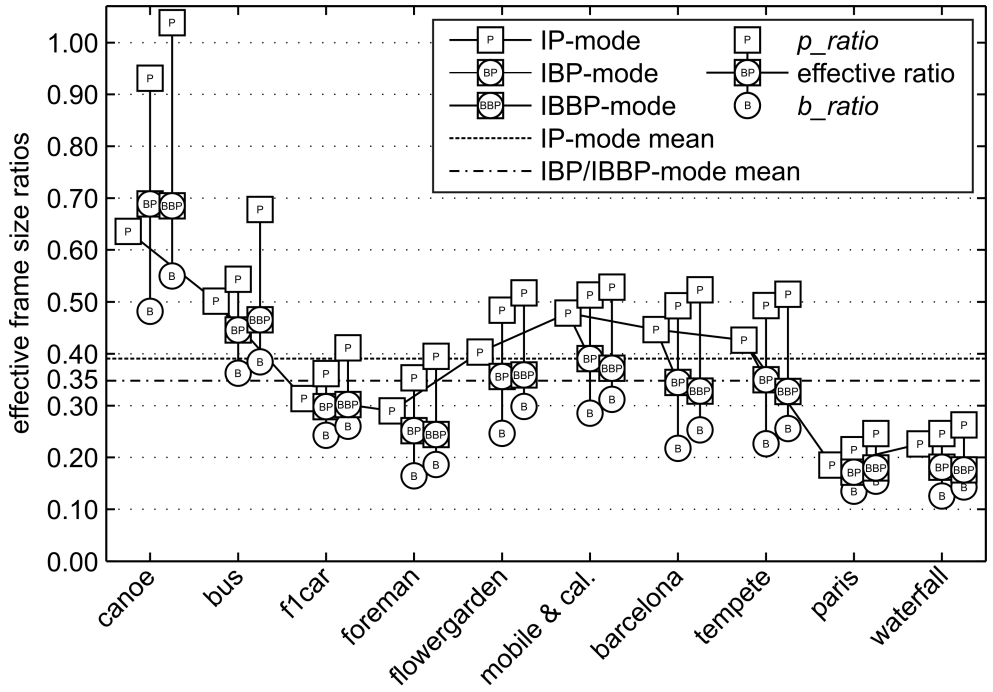


Figure 3.11: MPEG-2 bi-directional prediction mode frame size ratios. ($N_{GOP} = 12$, $PSNR \approx 35\ dB$, IBP and IBBP mean values are identical).

3.3.5 Coding mode analysis summary

In the previous sections the various temporal prediction modes were analyzed in detail by manual adjustment of quantizer values for various operating points. In this section the individual results are summarized and verified against the measured *PSNR* image quality.

For the analysis the averaged bit rate measures and *PSNR* measures are displayed in figure 3.12. The H.264 codec has significantly larger gains for the IP-coding mode and performs about $2.5\ dB$ better than the MPEG-2 codec in medium bitrate applications which is equivalent to approximately 38 percent data rate savings in the $35\ dB$ operating point. The comparison is however not completely satisfactory because the H.264 codec uses RD-optimization whereas the MPEG-2 codec implementation is not RD-optimized. In the case of intra coding, the H.264 standard performs only $10\ \%$ better in terms of data rate (without deblocking filter) than MPEG-2.

Seen from the large scope, the gain of coding modes with multiple reference frames appears less significant than the gain of simple IP-coding compared with intra coding. For the precise analysis of H.264 temporal prediction modes, the linearized rate-distortion function at the $35\ dB$ operating point was calculated by using the average slope of the RD-function in the range of 30 to $40\ dB$. This way, the small variations in quality which are caused by the fixed quantization steps can be compensated.

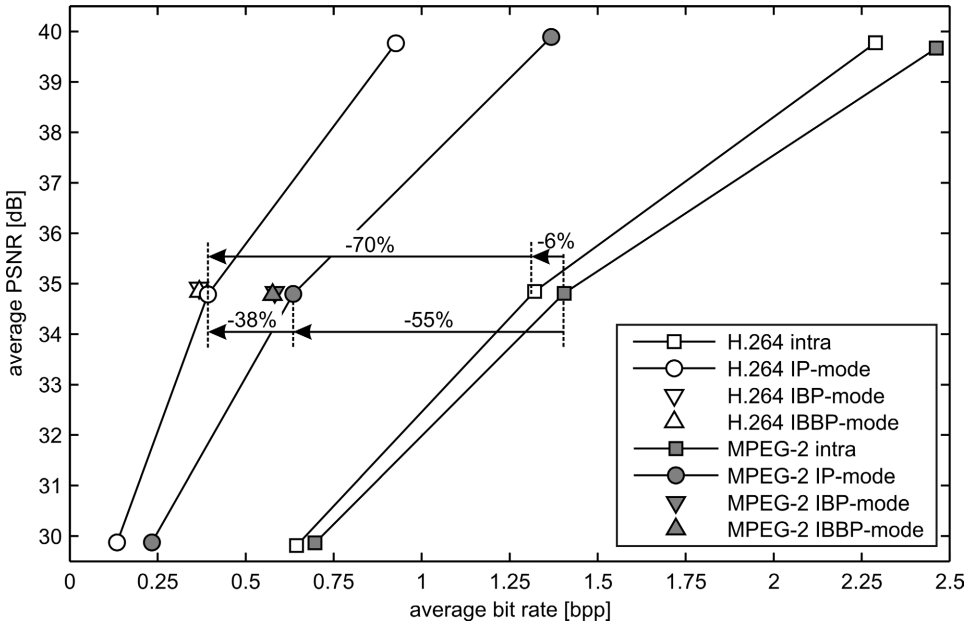


Figure 3.12: H.264 and MPEG-2 rate distortion curves.

The results shown in figure 3.13 are generally similar to the results without quality correction of the previous sections. For H.264 the forward prediction modes with two (IP2) and three (IP3) reference frames lead to 4.9 and 6.8 percent rate savings compared with the simple forward prediction mode. If the number of reference frames is further increased, only little improvements of 1.1 % and 0.7 % are measured for each additional reference frame.

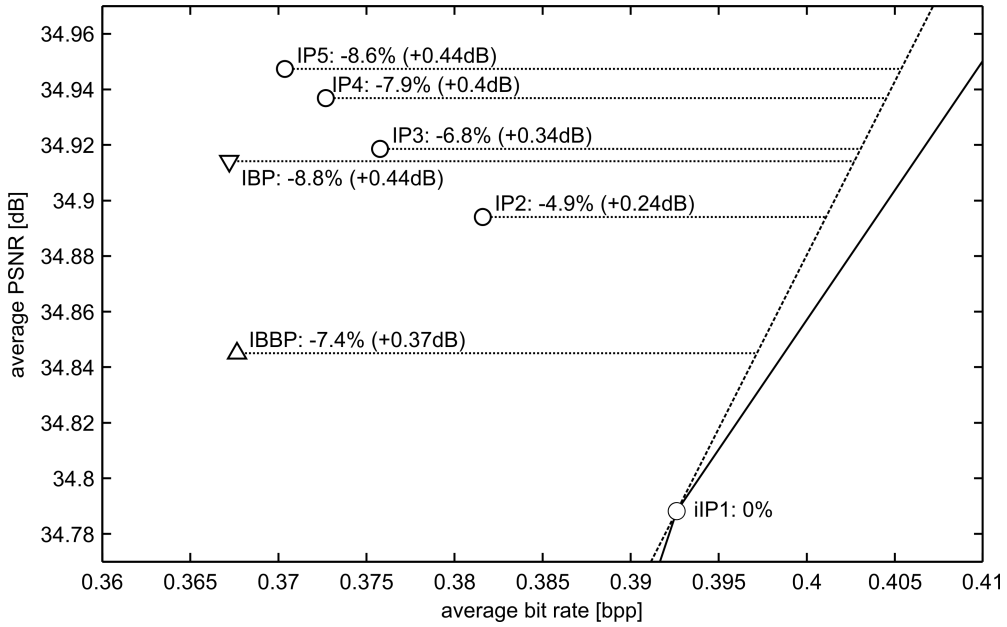


Figure 3.13: H.264 coding modes quality analysis with respect to IP-coding using only one reference frame. Dashed lines indicate the linearized RD-curve of the $35\ dB$ operating point, dotted lines indicate the projection on this linearized operating point.

The IBBP coding mode performs only slightly better than the IP3 mode and IBP achieves the best results with 8.8 percent rate savings compared with IP-coding. For MPEG-2 the results are 8.7 % bit rate savings for IBP mode and 9 % bit rate savings for IBBP mode (results not shown).

Summarizing the above discussion it can be concluded that the H.264 forward prediction modes with three to five reference frames deliver the same quality as the bi-directional prediction modes with two reference frames. Concerning computational complexity issues, the largest quality gain with the forward prediction can be achieved by adding a second reference frame.

Concerning the selection between MPEG-2 and H.264 video standards it can be concluded that the H.264 standard without deblocking performs 6 % better than MPEG-2 in intra mode and 38 % better in IP-mode if all macroblock partitioning modes are allowed.

3.4 Localized data rate analysis

In the previous section the frame-level data rate of video streams was analyzed with respect to different coding modes and quality levels. The simulations clearly indicate that rate variations on a frame level within the same frame type and video sequence are relatively small and change slowly over time. There is however a significant difference in the size of intracoded and predicted frames which results in high buffering latencies (chapter 4).

For achieving very low latency video transmission with a constant data rate, the rate distribution on a sub-frame level is examined in this section. The analysis of macroblock and slice level rate variations forms the basis of low-latency buffering concepts presented in section 4.5.

3.4.1 Macroblock data rate variations

A single macroblock forms the smallest unit for the rate control and rate distribution analysis. In figure 3.14 a graphic representation of the probability distribution functions for the data rate on macroblock level is shown for selected video sequences. The PDFs include the average data rates $\bar{x}$ and $2\bar{x}$ for reference. It should be noted that characterizing the PDFs with standard deviation does not lead to comparable results as the distributions have very different shapes.

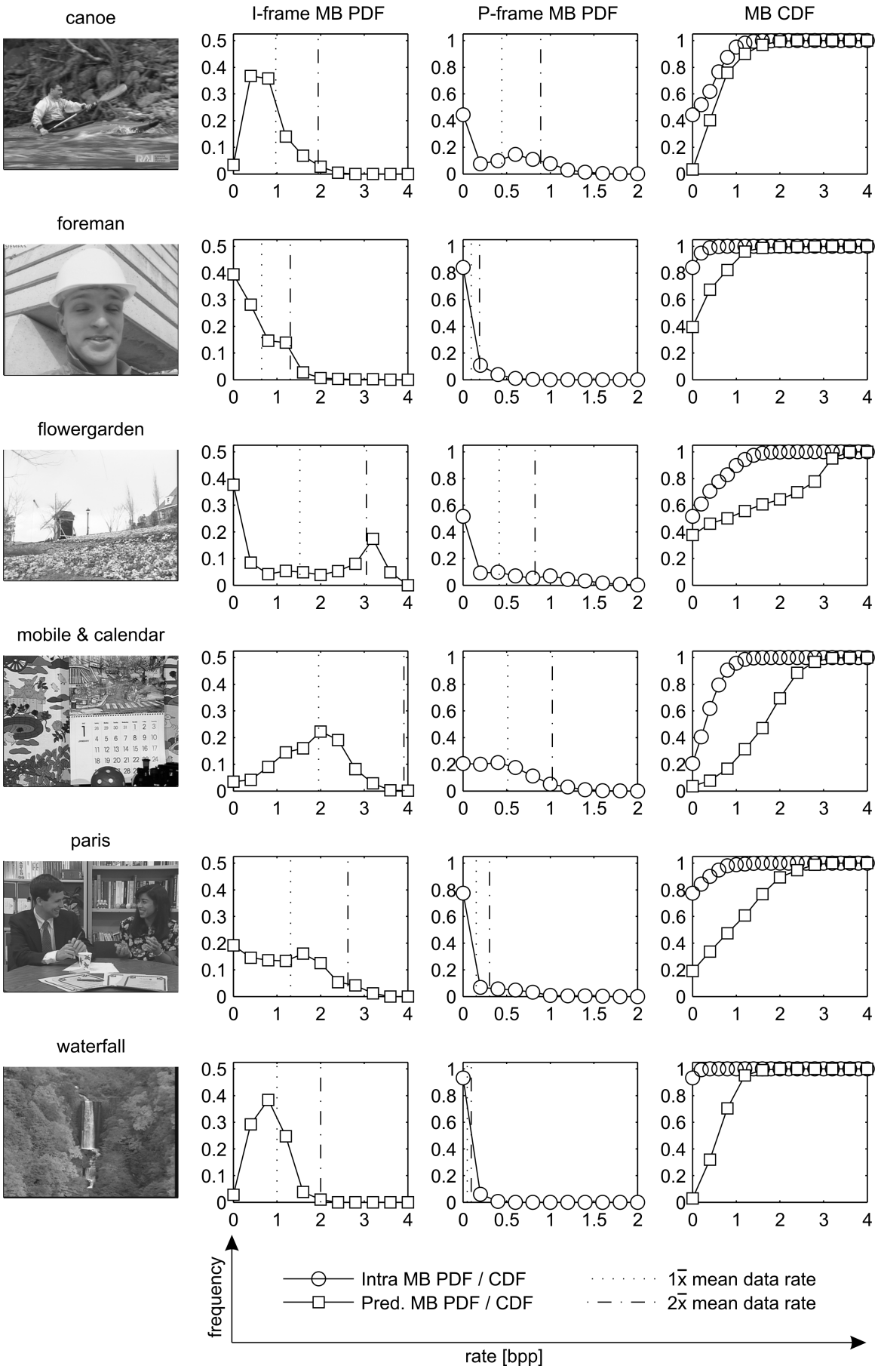


Figure 3.14: Macroblock data rate PDFs and CDFs for selected video sequences (samples taken from the first three frames, CIF format, $PSNR \approx 35\ dB$).

Concerning the intra frame macroblock rate distributions, it can be seen that there exists a strong correlation between texture complexity and data rate. Considering the PDFs of the *waterfall* and *canoe* sequence, it can be concluded that detailed homogeneous texture leads to concentrated energy and rate distribution close to the average frame data rate. Furthermore, none of the macroblocks is coded at zero data rate.

On the other hand, the *foreman* and *flowergarden* sequences have large regions with insignificant texture, resulting in about 40 % of macroblocks coded at a very low data rate of less than 0.16 bpp (approx. 40 bits per macroblock). The *flowergarden* sequence also shows a second peak above three bpp which is caused by the random-noise like flowerbed texture at the lower part of the image. A third group of characteristics can be found at the *mobile & calendar* and *paris* test sequences. Both sequences have a balanced distribution of texture resulting in flat PDFs for the intra coding modes.

Comparing the predicted frames and the intracoded frames it can be seen that the PDFs of the predicted frames are shifted towards lower data rate and the peak of the rate distribution is usually below 0.8 *bpp* (20 bits per macroblock). Only the *canoe* and *mobile & calendar* sequences have significant peaks in the 0.5 *bpp* region which are likely to be caused by discontinuities of motion vectors in both sequences.

For the system design point of view it is also interesting to estimate the peak performance required for the entropy encoding and decoding stages of a video codec. This can be characterized by the percentage of macroblocks which exceed multiples of the average data rate for which the system is designed.

Table 3.4: Characteristic of "large" macroblocks exceeding multiples of the average data rate. (Note: values are calculated from the raw data, not the coarse CDFs shown in figure 3.14).

	Intra frames			Predicted frames		
sequence	$\bar{x}$ **[bpp]**	**P(x>2** $\bar{x}$ **)**	**P(x>3** $\bar{x}$ **)**	$\bar{x}$ **[bpp]**	**P(x>2** $\bar{x}$ **)**	**P(x>3** $\bar{x}$ **)**
canoe	0.97	3.5 %	0.0 %	0.44	17.8 %	2.9 %
foreman	0.65	12.9 %	1.2 %	0.09	17.6 %	9.8 %
flowergarden	1.52	25.8 %	0.0 %	0.41	21.9 %	9.3 %
mobile & calendar	1.95	0.1 %	0.0 %	0.51	8.2 %	0.6 %
paris	1.31	6.9 %	0.0 %	0.15	19.4 %	13.8 %
waterfall	1.00	1.0 %	0.0 %	0.05	16.1 %	11.1 %

In the case of intracoded frames the average data rate lies in the range 0.65 and 1.95 bits per pixel for a constant quantizer value. Only the *foreman* and *flowergarden* sequence have significant amounts of very complex macroblocks

which exceed the double average data rate. As the average data rate of the *foreman* sequence is quite low, only the PDF peak at three bpp of the *flowergarden* sequence may be considered as a difficult issue for resource limited codecs. In practical CBR implementations the high average data rates of *mobile & calendar* and *paris* are less difficult to handle as their PDFs are quite flat resulting in even workloads for the entropy coding stages.

Although the predicted frames have smaller average data rates, the rate variations are more significant. Between $10\ \%$ and $20\ \%$ of the macroblocks exceed the double average data rate. It is also remarkable that the low-complexity video sequence *paris* has the largest amount of macroblocks exceeding $3\,\bar{x}$.

3.4.2 Slice resolution and randomized data rate analysis

In order to achieve a constant data rate it is necessary to average out the local rate variations over a defined time interval. As indicated in the discussion of section 3.3.3 and figure 3.3, the frame-level data rate is relatively constant over time for each type of frames. On the other hand, macroblock data rates from zero up to 900 bits were observed within the same data stream. The slice-level averaging over 22 macroblocks, e.g. one slice in CIF resolution, is a first approach for finding bounds of rate distributions.

In figure 3.15 the slice level data rates of six video sequences are shown for the first intra and predictive coded frame. All intracoded frames use the same quantizer value which results in an average video quality of $35\ dB\ PSNR$. The quantizer of predicted frames is adjusted to match the average intra quality of the video sequence.

For the intra coding mode, the average data rate is in the range of one to two bits per pixel. The localized slice averages show however significantly larger variations, ranging from less than $0.5\ bpp$ to $3.5\ bpp$ peak rates for the *flowergarden* sequence. A detailed analysis of the individual sequences shows that the *foreman* and *waterfall* video sequences have the smallest variations due to the uniform distribution of texture in the vertical image dimension. The sequences *paris* and *mobile & calendar* show slightly stronger variations for the intra coding mode but the strongest effects of local rate variations occur in the *flowergarden* and *canoe* sequences. In these two sequences the amount of texture information changes significantly in the vertical dimension, resulting in long sequences of small macroblocks in the upper frame section and sequences of large macroblocks in the lower frame section. Concerning the buffer delays which will be discussed in more detail in section 4.5, these two sequences represent boundary cases which must be taken into account for the low-latency transmission system design.

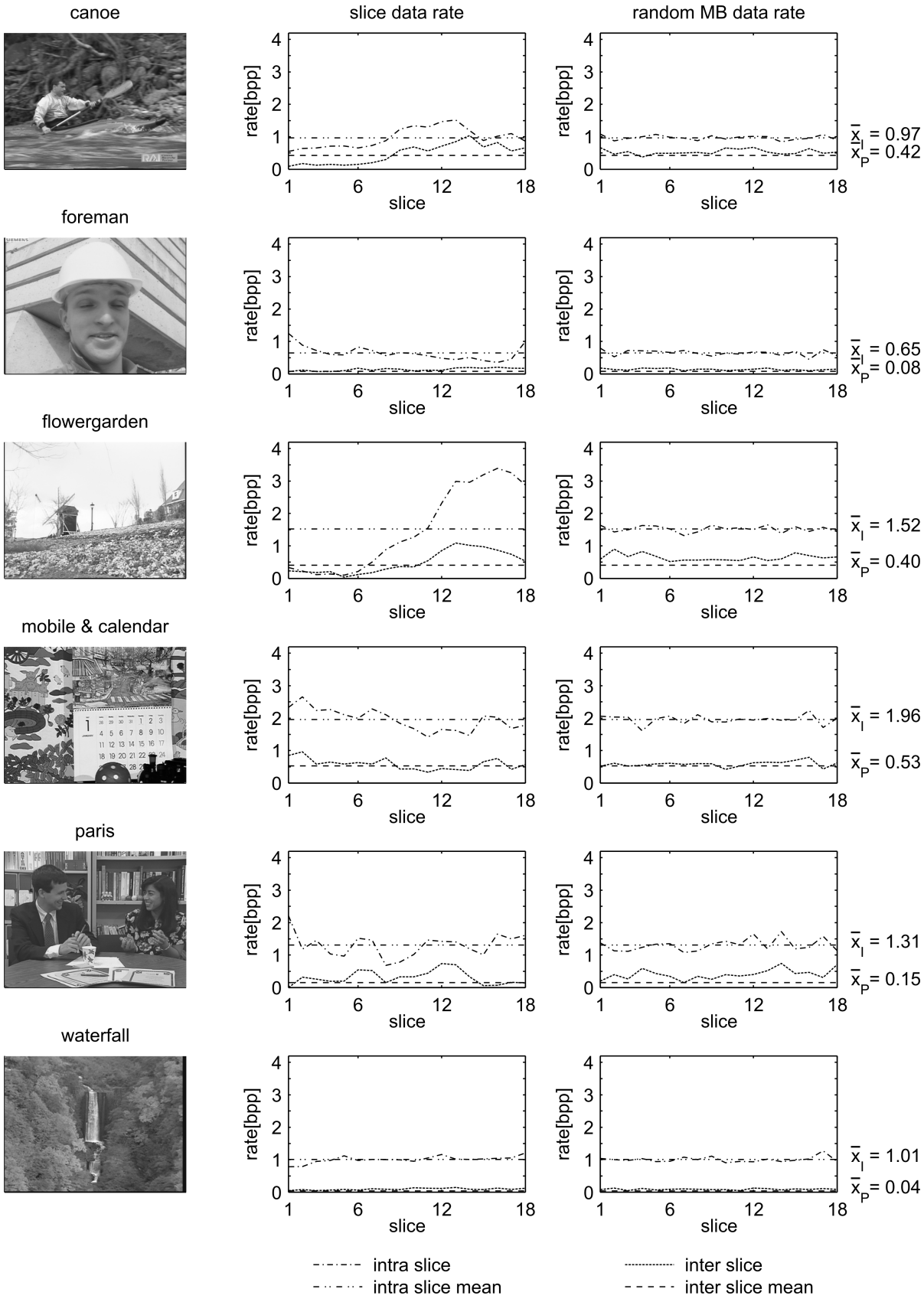


Figure 3.15: H.264 slice level rate variations with row-wise slices and randomized slices ($PSNR \approx 35\ dB$).

Concerning the predicted mode, the slice data rate distributions are similar to the intra coding mode results although a different behavior could be expected if the macroblock data rate PDFs are considered. The results in figure 3.15 clearly indicate that the local rate variations between intra and predictive mode are strongly correlated, especially for the *canoe* and *flowergarden* sequence. However, in figure 3.14 all sequences except *mobile & calendar* show quite similar PDFs in the predictive mode. Hence, it can be concluded that it is not possible to make direct conclusions from the macroblock PDFs to the slice level data rates.

On the left side the simulation results for a random access scheme are shown for 18 groups of macroblocks with randomly selected macroblocks of the whole frame. The strong rate variation of the slice level data rate is leveled out to the largest extent, resulting in an almost constant data rate. This strategy would ease the buffering for low-latency applications, but it would introduce additional delays for the randomized reordering of macroblocks as discussed later in section 4.4.2.

4 Latency Analysis for Standard-mode Video Compression Systems

4.1 Generalized video transmission model

In the following sections a basic model for compressed real-time video transmission is introduced and refined with more detail. The standard configuration one-way video transmission system as illustrated in figure 4.1 consists of a video source, the video encoder, the fixed-rate transmission network, the decoder and a video display. Compared with an uncompressed (e.g. analog) transmission where the video source directly interfaces with video display, the video encoder, network and decoder introduce additional latencies which should be minimized.

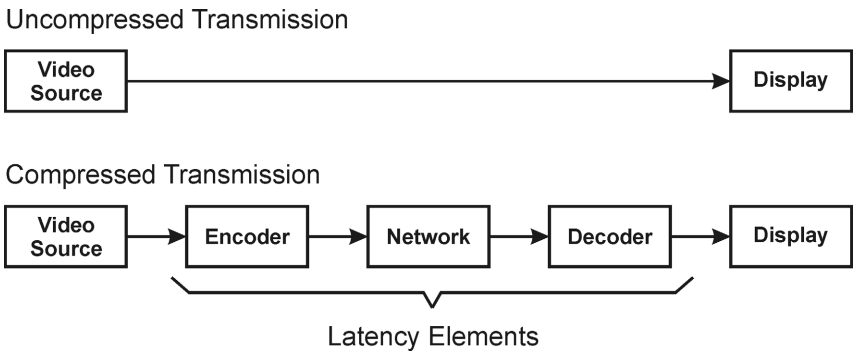


Figure 4.1: Uncompressed and compressed video transmission systems.

For further analysis a more detailed latency model as shown in figure 4.2 is introduced. On the encoder side the latency-critical elements can be identified as the input pixel buffer (block capture), the frame reordering buffer, the encoder processing and the encoder output buffer. The decoder comprises an input stream buffer, the decoder processing and the output pixel buffer (block output) as latency-relevant components. Concerning the low-latency optimization of such video transmission systems, splitting the model into three groups proved practical: algorithmic encoder latency, transmission latency and algorithmic decoder latency. Later discussions will show that algorithmic and transmission latencies can be optimized with minimal effect on each other.

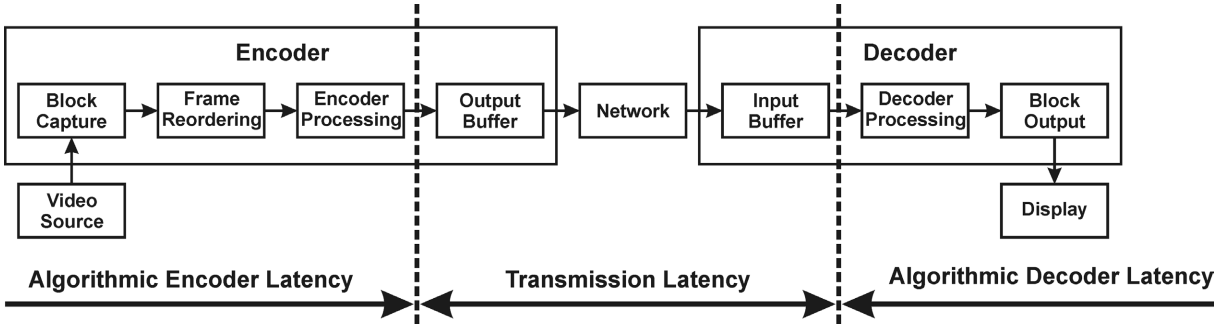


Figure 4.2: Detailed video transmission latency model.

According to figure 4.2, the system delay D_{sys} which should be minimized can be calculated from the sum of individual latencies:

$$D_{sys} = D_{cap} + D_{reorder} + D^{e}_{proc} + D^{e}_{buff} + D_{net} + D^{d}_{buff} + D^{d}_{proc} + D_{out} \tag{4.1}$$

The **block capture** device is introduced for modeling the effects of continuous-time video sampling. It is assumed that the video source samples and transmits the image data from top left to bottom right as a continuous digital stream. The block capture device collects the incoming pixel data in a pixel buffer until enough data is available to start the processing. It introduces a delay corresponding to the time interval between the sampling of the first and the last pixel of a processing block.

The corresponding **block output** device in the decoder collects the pixel data of the decoder processing block until all samples are available for displaying.

In many software-based implementations with operating system support, D_{cap} and D_{out} are equal to one frame period because the operating system routines usually only provide frame-level accurate capture and display synchronization. In delay-critical systems it is also possible to reduce the capture and output delay to one slice period ($D_{cap} \approx 0.05\ T_{frame}$). However, the shorter synchronization intervals increase the peak-load computational requirements for the encoder and decoder.

Analyzing the **transmission latency** requires joint consideration of the encoder output buffer, the transmission channel and the encoder input buffer. In contrast to other publications which focus on variable bitrate transmission, e.g. [48,49], it is the goal of this thesis to achieve CBR operation with a constant network load. Hence, the encoder output buffer must smooth the data rate variations and the encoder input buffer must provide sufficient capacity to ensure continuous playback. A larger buffer generally relaxes the rate control requirements and introduces longer delay. In the following discussions the network delay is generally neglected ($D_{net} \approx 0\ T_{frame}$).

Figure 4.3 shows an example process-scheduling diagram for a generic intra-frame video transmission system with a processing block size of one frame and a

constant bit rate at frame level. In this scenario the encoder processing starts immediately when the whole frame is captured. In the case of intra-frame compression the frame reordering can be omitted ($D_{reorder} = 0\ T_{frame}$).

As described in section 3.4.2 the data rate of an intra-coded frame frequently shows a non-uniform distribution, which results in non-linear filling of the encoder output buffer. Therefore, if no restrictions on the data rate distribution within a frame are imposed, the complete image must be stored in the encoder output buffer before the transmission starts. During transmission over a CBR channel, the encoder buffer will be emptied linearly while the decoder buffer will accumulate the transmitted data.

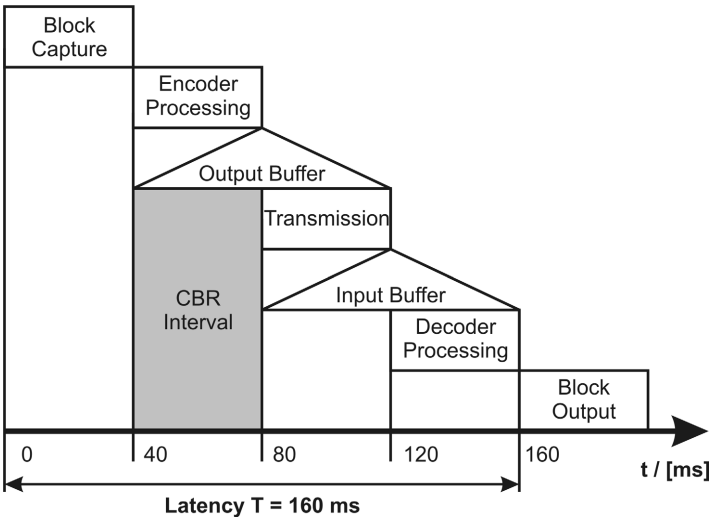


Figure 4.3: Process-scheduling for intra-frame compressed video transmission (progressive scan 25 frames per second, $T_{frame} = 40\ ms$).

The decoder receive buffer stores all incoming data until it is processed by the decoder. If the distribution of data within the frame is unrestricted, the decoder must delay the start of decoding until it has received the complete frame data. Under the real-time constraints of a frame-based processing scheme, the frame reconstruction must be completed after one frame period and the image can be displayed from the output pixel buffer. Considered as a whole, this scenario introduces four frame periods latency from the beginning of frame capture to the beginning of displaying the decoded frame. This results in $160\ ms$ delay for progressive $25\ Hz$ video signals and $80\ ms$ for interlaced SDTV signals.

4.2 Algorithmic encoder and decoder latency

Generalized encoder model

On a data flow level, the generalized video encoder shown in figure 4.4 can be broken down into the block capture device, the frame reordering buffers, the inner encoder processing block and the output stream buffer. The inner encoder comprises the well known hybrid encoding scheme with temporal inter-frame-prediction. This encoder concept is valid for intra coding standards like JPEG2000 (without reconstruction and predictor) as well as MPEG standards with forward and backward temporal prediction. In the following section the algorithmic latency for optimal low-latency processing is analyzed.

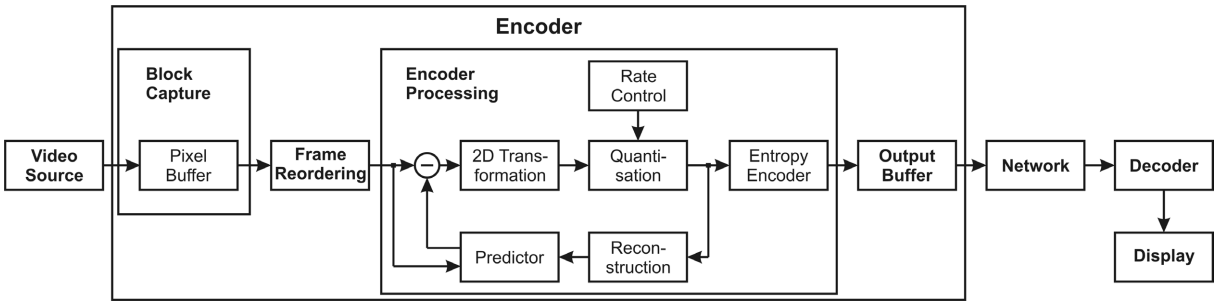


Figure 4.4: Generalized encoder model.

Block capture latency

The commonly used video coding standards allow processing block sizes significantly smaller than one frame. In this way the latency as well as the memory requirements for the encoder can be reduced.

The **processing block size** is defined as the size of the frame segment which is available in the block capture device for immediate processing. The resulting **block capture latency** D_{cap} can be calculated from the number of lines N_{block} of the processing block size and the duration of a video scan-line $t_{line} = 1/f_{line} \approx 64\ \mu s$ for SDTV:

$$D_{cap} = N_{block} \bullet t_{line} \tag{4.2}$$

As the scan-line frequency of the video source usually cannot be changed, the block capture latency is proportional to the chosen processing block size. For the common MPEG video standards the lower bound on the processing block size is determined by the macroblock height of 16 lines. Considering pure progressive or interlaced coding modes, this results in a block capture latency of approximately

$$D_{cap} = T_{slice} = 16 \bullet 64\ \mu s \approx 1\ ms = 1/40 \bullet T_{frame} \tag{4.3}$$

for SDTV signals.

However, an analysis of the video codec reference source code [20,21] shows that non-optimized implementations usually load a complete frame into the memory before processing starts. This results in a capture latency of one frame period which makes in-depth re-organization of the reference code necessary for low-latency video processing.

For the interlaced frame pictures coding mode described in section 2.3.6.2, it is necessary to capture the first field and the second field up to the current macroblock. This results in block capture latency of $D_{cap} = 20\ ms + 8 \bullet 64\ \mu s \approx 20.5\ ms$ for SDTV signals. The processing block size is however still equal to one row of macroblocks, as the other lines of the first field are not available for immediate processing. System integration aspects of this coding mode will be discussed in section 4.4.5.

The block capture delay of the wavelet-based JPEG2000 standard (PCRL coding mode) is defined by the tile height. For a practical processing block size (tile size) of 32 by 128 pixels this results in a latency of 32 lines of video (SDTV: $D_{cap} = 32 \bullet 64\ \mu s \approx 2\ ms$).

Frame reordering latency

For implementing bi-directional temporal prediction modes, it is necessary to have the past and future reference frames available in the encoder pixel buffer before encoding the enclosed B-frames. A detailed analysis (section 4.4.4.) of this effect reveals, that the delay is introduced either in the encoder or in the decoder, depending on the coding mode of the frame. As the overall latency for all frame types is constant throughout the system, the frame reordering latency $D_{reorder}$ is counted only once in the encoder block.

The bi-directional coding mode generally introduces additional delays of multiple frame periods which makes it unsuitable for low-latency applications. In the case of intra- and IP-coding modes, the frame reordering is inactive and $D_{reorder} = 0$.

Encoder processing latency

In a real-time implementation of a video encoder it is important to minimize the hardware and memory requirements. It is therefore a straightforward solution to process sections of the image, i.e. macroblocks or tiles, in a sequential, pipelined manner. It should be noted that for intra as well as predictive coding modes, the encoder processing can start immediately when the last line of a macroblock is captured.

For a pipelined hardware implementation, the functional blocks for motion estimation, prediction, transformation, quantization, entropy coding, inverse transformation and reconstruction can work in parallel on different successive

image blocks if there is no direct data dependency feedback between adjacent blocks of the same frame[2]. For MPEG-2 at SDTV resolution, $25 \cdot 1620 = 40500$ macroblocks have to be processed every second, so every macroblock processing step is allowed to take at maximum $25\ \mu s$. Relevant steps for the encoding latency are the prediction, transformation, quantization and entropy coding resulting roughly $D^e_{proc} = 4 \bullet 25\ \mu s = 0.1\ ms$ for pipelined macroblock processing.

For a programmable DSP-based implementation it is also reasonable to use macroblock-based processing instead of frame-based processing as it reduces memory requirements and memory accesses significantly (section 6.2). Assuming a single CPU implementation without hardware support it is obvious that the maximum allowable CPU time for encoding one macroblock is $D^e_{proc} = 1/40500\ s \approx 25\ \mu s$. This also corresponds to the minimum processing delay.

In comparison with the $1\ ms$ block capture latency, the macroblock processing latency of an optimized encoder is quite insignificant. For the convenience of implementation, workload averaging and rate-control issues, it may be advantageous to perform the encoding on image segments corresponding to the block capture size. This approach results in a processing latency which is equal to the block capture latency.

It should be noted that the encoder processing delay can be reduced by increasing the processing speed, i.e. using a higher clock frequency. This is only possible because the block capture device ensures the availability of all input data for the processing stage. From an efficiency point of view it is however advantageous to achieve a constant workload over time in all processing units [50].

Decoder processing and block output latency

Similar to the encoder scheduling, the decoder processing delay can be optimized for low-latency applications. As illustrated in figure 4.5 only the entropy decoding, inverse quantization, inverse transformation and addition of the motion-compensated prediction are performed in the time-critical forward path. Hence, for a pipelined and macroblock-based implementation, the processing latency is similar to the encoder delay of $D^d_{proc} = 0.1\ ms$. Increasing the decoder clock frequency can reduce the decoding delay in the same manner as discussed for the encoder. The block output size should be set equal to the processing block size to achieve zero block output delay $D_{out} = 0$.

[2] Valid for all MPEG-2 coding modes as only the temporal prediction refers to pixel data of reconstructed frames. However, for the H.264 intra prediction, the preceding neighboring macroblocks of the same frame are needed to form the intra prediction of the current macroblock.

If the processing block size of the decoder is chosen larger than or equal to the block output size, D_{out} becomes zero because all decoded frame data can be displayed immediately when the decoder processing is finished. If the block output size is larger than the processing block size, the block output device must delay the display output until the whole block was decoded.

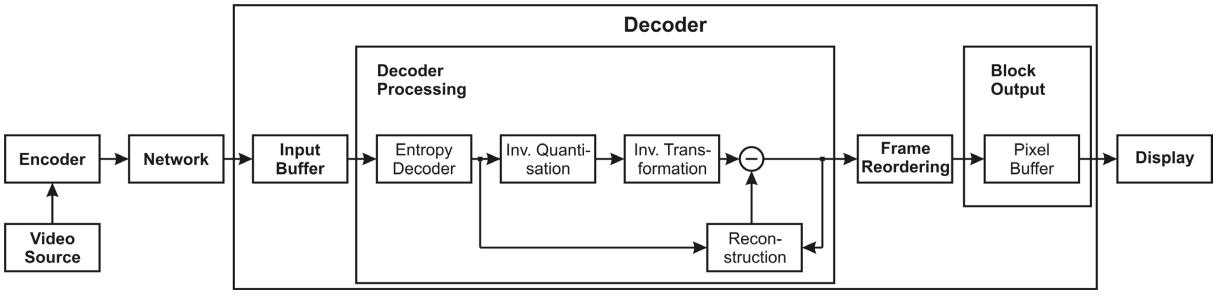


Figure 4.5: Generalized decoder model.

4.3 Transmission latency

The reduction of buffer delays is a key issue for low-latency applications, and a detailed analysis of optimized buffering concepts for constant bitrate operation (CBR) will follow in section 4.5. A related discussion on temporal rate variations can be found in [49] with a focus on variable bitrate (VBR) encoder operation.

Achieving a constant data rate from a stream with varying compressed frame sizes is only possible if the variations are averaged out over a time interval significantly larger than one frame period. Practically, the data of big I-frames must be spread over the following smaller P and B-frames in the GOP. Hence, the encoder output buffer averages peaks in the data rate and the decoder input buffer ensures the availability of input data for the decoding process in a real-time system.

In the following discussions it is assumed that the transmission channel has a given maximum available bandwidth which may not be exceeded under any circumstances. Ideally the compressed video data stream has a constant data rate R expressed in bits per second using the maximum available bandwidth.

The rate control of the encoder must avoid buffer over-runs and under-runs which would result in either data losses in the encoder buffer or decoder errors caused by late arrival of data. To ensure proper real-time operation the decoder buffer is modeled in the encoder with the virtual buffer verifier (VBV). Furthermore, maximum buffer sizes for the decoder are defined within the standards to ensure encoder-decoder compatibility. When the video transmission starts, the decoder waits until a certain buffer occupancy is reached before it starts decoding the stream. The MPEG-2 as well as H.264 standards also define that the decoder may

not start decoding a frame before it is received completely. It is theoretically possible to overrun this rule if the required data is available in time at the decoder, e.g. a CBR transmission with a constant data rate below frame level. For MPEG-2 (MP@ML) standard parameters a maximum buffer size of $16 \cdot 116\ kbit = 1.8\ Mbit$ is allowed, resulting in 0.5 seconds delay at $3.6\ Mbit$ data rate [18].

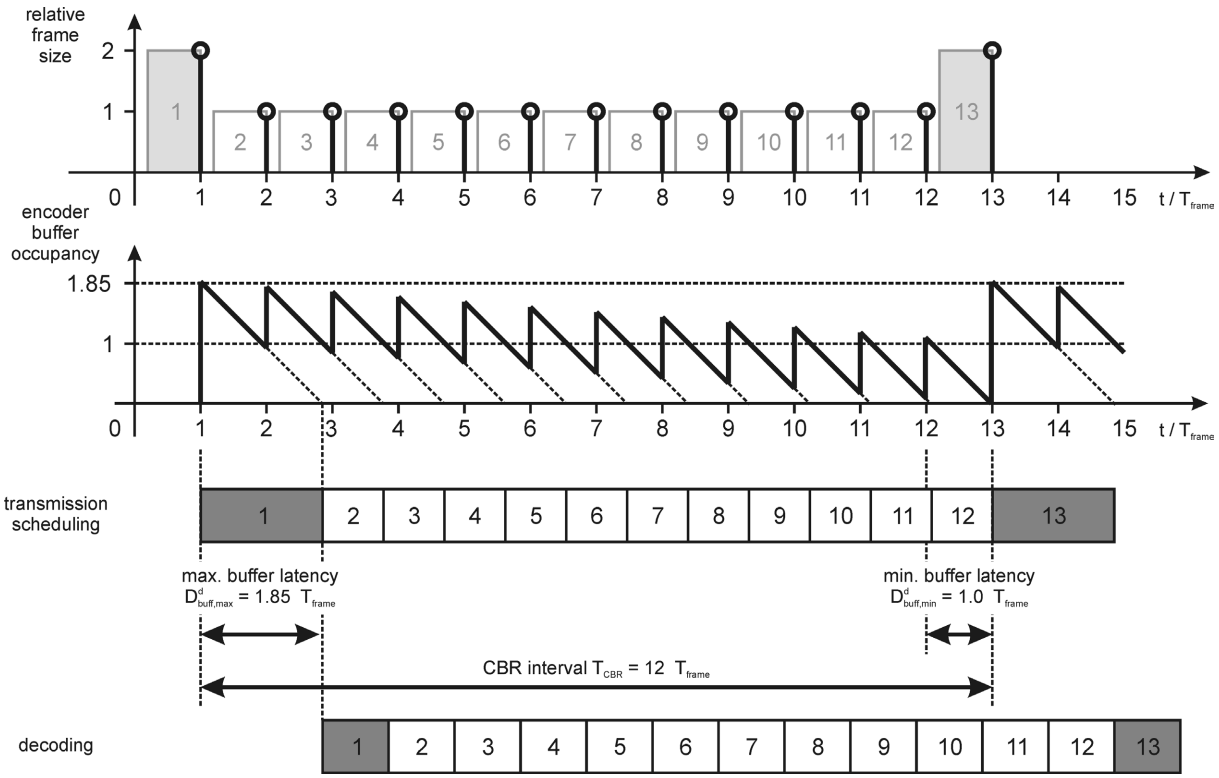


Figure 4.6: Encoder buffer occupancy and transmission scheduling for IP-coding.

An example scenario for the encoder output buffer in IP coding mode is illustrated in figure 4.6. The analysis is based on the assumption that the initial I-frame at the start of GOP has twice the data rate of the following P-frames as shown in the upper part of the figure. As the data distribution within the frame is unknown and unrestricted, the input of the encoder buffer is modeled dirac-like at the end of its frame period for frame-based processing. In the case of optimized slice-based processing, the frame-dirac modeling introduces an additional $D^e_{buff} = 1\ T_{frame}$ buffer delay in the encoder. Hence, the start of transmission is delayed by one frame period after placing the data of the first slice in the output buffer.

For the constant bit rate per GOP (GOP-CBR) operation mode, the average transmission rate R is set to the average data rate of the GOP resulting in equivalent buffer occupancy at the beginning and the end of the GOP. From these conditions a minimum decoder buffer size $B_{min} = 2 \bullet 12/13 \bullet R \bullet T_{frame} = 1.85 \bullet R \bullet T_{frame}$ can be calculated from the size ratio of the initial I-frame and successive P-frames. Hence, the decoder must delay the start of decoding by $1.85\ T_{frame}$ after receiving the first bits of frame data assuming

that the rate distribution within the I-frame is unrestricted. The encoder buffer occupancy graph of figure 4.6 shows that this buffer requirements for the whole video sequence are fulfilled as long as the frame size ratio and the I-frame period do not change. As indicated in [49] the buffer delay in GOP-CBR operation for the IP coding mode is bounded by T_{GOP} if a static video scene concentrates $100\,\%$ of the data rate in the intra-coded frames.

According to the leaky bucket model, which will be discussed in detail in section 4.5.1, the encoder and decoder buffers are equally sized and introduce a constant delay D_{buff} in the system. However, the portions of encoder buffer delay D^{e}_{buff} and decoder buffer delay D^{d}_{buff} vary dynamically.

The **buffer latency** D_{buff} is defined as the time interval from the moment of (dirac-modeled) placing of picture data in the encoder output buffer until all frame data is removed from the decoder receive buffer.

For a given constant bit rate R [bit/s] of the transmission channel, the buffer latency D_{buff} can be calculated by the ratio of the buffer size B [bit] and the transmission rate R:

$$D_{buff} = D^{e}_{buff} + D^{d}_{buff} = \frac{B}{R} \tag{4.4}$$

In the given example, the initial decoder buffer latency $D_{buff} = 1.85\ T_{frame}$ is equal to the transmission time of the I-frame over the CBR channel. It also defines the earliest possible start for decoding if no limitations on the rate distribution within a frame are imposed. Although the decoder buffer latency decreases to $1\ T_{frame}$ for the first P-frame of the GOP, this does not speed up the displaying because the encoder buffer will compensate for a constant frame rate with an additional $0.85\ T_{frame}$ buffer delay.

For the example shown in figure 4.6, two theoretical options exist for reducing the latency:

1. Encoding the I-frames at a low quality with the size of P-frames would reduce the transmission latency to $1\ T_{frame}$. However, the low-quality I-frames would generate periodical quality variations at a low-frequency of $2\ Hz$. This kind of quality variations would be very annoying for the viewer and cannot be accepted in a commercial product.

2. Increasing the available channel data rate to the peak data rate of the I-frames would reduce the transmission latency to $1\ T_{frame}$. For the given example, the average data rate of the video stream would be $54\,\%$ of the channel data rate, hence $46\,\%$ of the channel capacity would be lost.

Based on these considerations and the simulation results presented in chapter 3, a time interval in which the encoder generates a constant amount of data can be identified for each coding mode under the assumption that the rate control algorithm only compensates for long-term rate variations.

The **CBR interval** T_{CBR} is defined as the shortest time interval in which the encoder generates a constant amount of data B_{CBR}. Considering the simulation results of chapter 3, three different CBR modes can be distinguished:

CBR Mode	CBR interval	applicable coding modes
Block-CBR	$T_{CBR} = T_{block} < T_{frame}$	randomized intra coding mode
Frame-CBR	$T_{CBR} = 1\ T_{frame}$	Intra coding mode, frame-CBR IR
GOP-CBR	$T_{CBR} = T_{GOP} = N_{GOP} \bullet T_{frame}$	IP, IPB, GOP-CBR IR coding modes

Furthermore, two relevant cases of buffer management can be identified:

Unconstrained CBR: $D_{buff} = T_{CBR}$

Assuming a constant bit rate for a GOP, this scenario would introduce a $T_{GOP} = 12\ T_{frame}$ buffer latency in the system. The distribution of data within the GOP is however unconstrained, meaning that the I-frame can consume up to 100 % of the data rate within a GOP.

Constrained CBR: $D_{buff} < T_{CBR}$

This case corresponds to the example in figure 4.6 and imposes certain assumptions and restrictions on the distribution of data within the GOP.

The case $D_{buff} \geq T_{CBR}$ is not relevant for low-latency applications as it would introduce unacceptably large buffers in the system. It is on the contrary the goal of the analysis in section 4.5 to find reasonable lower limits for the constrained CBR operation mode to reduce the buffer delays considering relevant video test sequences.

Network Delay

As the analysis of network effects is not a major focus of this work, the network delay is generally neglected in the following discussions. A rough analysis of small-space networks like IEEE1394 [51] or automotive bus systems like MOST [52] reveals that these networks can generally provide very low delays in the range of few milliseconds if configured properly. Compared with the CBR buffer delays, the packetizing and transmission delays are quite insignificant. In contrast to this, large-scale networks like the internet can introduce significant delays and delay jitter which are investigated in detail in [48].

4.4 System latency analysis for standard coding modes

In this section the minimum latency of standard coding modes for CBR transmission is analyzed for optimized slice process scheduling. It is the goal of this analysis to identify the inherent algorithmic latency elements and buffers which define the minimum latency of the video transmission system. For the purpose of illustrating the buffer delays, simple assumptions of the rate distributions are made. A more detailed analysis of the buffer delay limitations will be presented in section 4.5.

The delay calculations for slice processing are based on a processing block size of one row of macroblocks in SDTV resolution. This results in capture and processing latencies of $T_{block} = 0.025\ T_{frame}$ calculated by the ratio of $1\ ms$ block capture latency at $40\ ms$ frame duration for $25\ Hz$ SDTV.

For the illustration of delay effects frame size ratios for P and B-frames are set to $p_ratio = 0.5$ (Eq. 3.6) and $b_ratio = 0.25$ (Eq. 3.7) respectively.

4.4.1 Intra coding mode

The intra coding mode is generally well-suited for low-latency video transmission over constant bit rate channels as all pictures generate approximately the same amount of data. Without temporal prediction, the coding efficiency is however quite bad resulting in at least twice the data rate of IP-coding schemes with forward temporal prediction. It may be a good tradeoff if transmission bandwidth requirements are less critical than computational complexity. Unlike all other coding modes discussed in the next sections, intra-frame coding is tolerant to scene changes as no temporal correlation of successive frames is exploited.

The latency analysis follows the timing scheme presented in figure 4.7, resulting in an overall latency of $2.05\ T_{frame}$ for slice processing. This analysis is based on the assumption of an inherent CBR interval of one frame as indicated by the simulation results in section 3.3.2. The unconstrained CBR operation requires delaying the start of transmission until the complete frame is encoded as well delaying the start of decoding until the complete frame was transmitted. Therefore, approximately 2.0 frame periods delay are introduced by the encoder and decoder buffers whereas the block capture and decoder processing introduce only $2 \cdot 0.025\ T_{frame}$ delay.

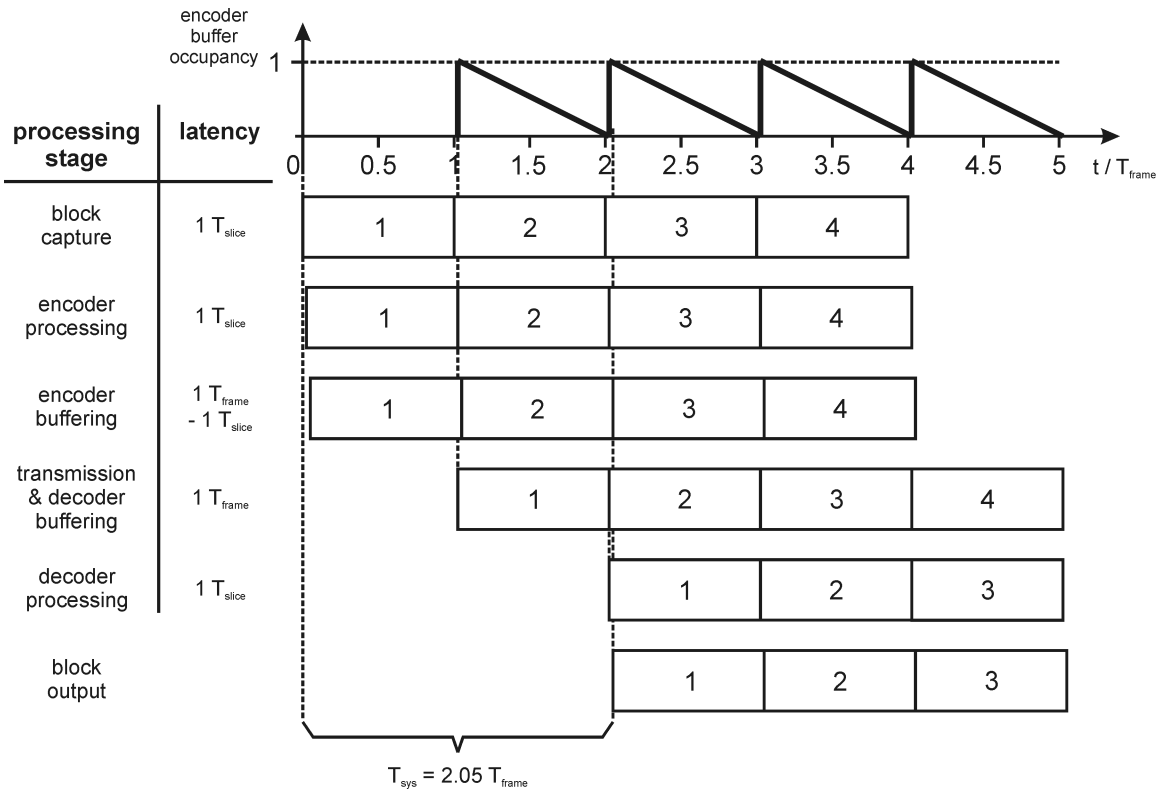


Figure 4.7: Process-scheduling for intra-coded video transmission with slice processing.

A detailed experimental buffer analysis in section 4.5.2 shows that careful adjustment of the encoder and decoder buffers allow an earlier start of transmission and decoding if the rate fluctuations in the frame are limited. For this reason the time reference $t = 0$ for the start of encoder buffering in figure 4.7 is set to the time when the data of the first slice is placed in the output buffer. In the case of unconstrained CBR operation, the start of transmission can start immediately when the last slice of the frame is encoded.

4.4.2 Randomized intra coding mode

In this section the system level latency analysis for a hypothetical randomized intra coding mode is presented. The randomization procedure is conceptually similar to the DVCAM-standard [25] for video cameras which requires constant bit rate operation for tape recording. Although other video compression standards like H.264 do not support macroblock randomization, this short discussion identifies the potential of this coding mode.

As indicated by the simulation results in section 3.4.2, macroblock randomization results in very good rate averaging allowing small buffer delays below $0.1\ T_{frame}$. The low buffer delays are, however, achieved at the cost of high algorithmic delays as illustrated in figure 4.8. On the encoder side, the start of macroblock randomization and encoding must be delayed until the whole frame was captured. Similarly, the displaying of the frame must be delayed until all macroblocks of the

frame were transmitted and decoded. Assuming a constant data rate for the randomized slices, the practical overall system latency is about 2.05 frame periods with a lower bound of 2.0 frame periods for the inherent algorithmic delays.

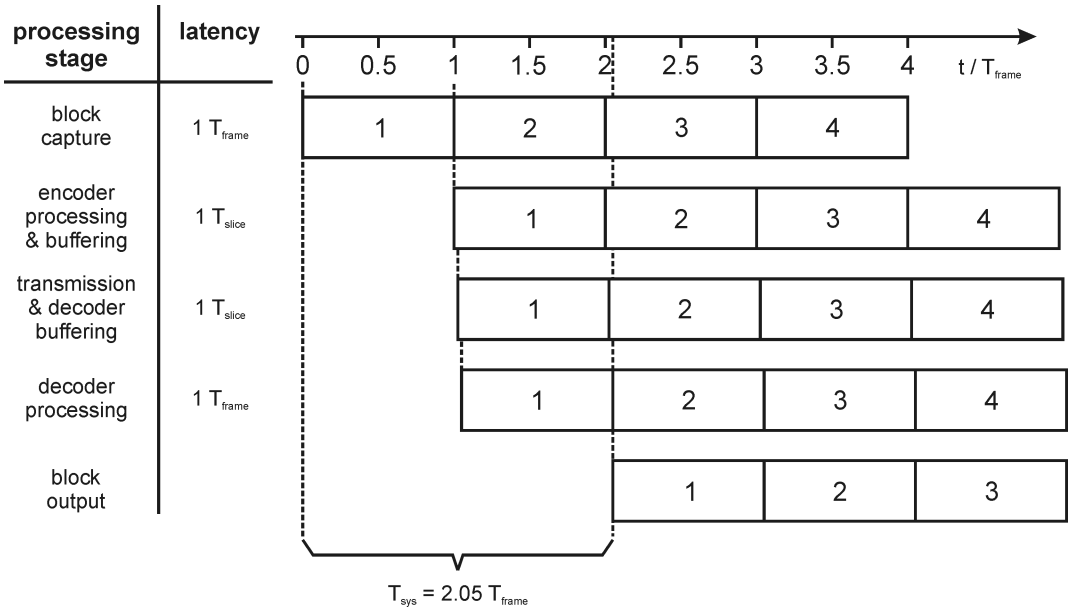


Figure 4.8: Process-scheduling for randomized intra-coded video transmission.

Considering these results, the macroblock randomization does not offer high potentials for low-latency applications compared with the normal intra coding mode which operates at the same efficiency.

4.4.3 Predictive coding mode

Figure 4.9 illustrates the timing analysis of a standard IP-coding scheme with a $50\ \%$ coding gain for predictive frames. As the intra-coded frame at the beginning of the CBR interval requires twice as much data as the following P-frames, approximately two time intervals must be allocated for its transmission. The minimum decoder buffer delay be calculated by:

$$D^{d}_{buff,IP} = \frac{1}{1+(N_{GOP}-1)\cdot p_ratio}\cdot N_{GOP}\cdot T_{frame}\text{, with} \tag{4.5}$$

N_{GOP} = number of frames in GOP (Eq. 2.2),

p_ratio = frame size ratio P/I frames (Eq. 3.6).

Equation 4.5 indicates that the buffer latency increases if the predictive coding gain $1/p_ratio$ increases or if the number of frames per GOP is increased. Hence, improving the temporal prediction as well as decreasing the frequency of I frames results in better coding efficiency at the cost of higher decoder buffer delays. Practical values for the p_ratio can be derived from figure 3.8. In the case of a low predictive coding gain with $p_ratio = 0.5$ and $N_{GOP} = 12$, the buffer delay evaluates to

$$D_{buff,IP} = \frac{1}{1+11\cdot\frac{1}{2}} \cdot 12\, T_{frame} = \frac{2}{13} \cdot 12\, T_{frame} = 1.85\, T_{frame} \tag{4.6}$$

In figure 4.9 it can be seen that the timing relaxes towards the end of the group of pictures allowing larger variations in the size of the P-frames. The system latency for the illustrated scenario is $2.90\ T_{frame}$ for implementations with slice processing. As indicated by the simulation results in section 3.3.3, the p_ratio can become significantly smaller resulting in higher delays for the constrained CBR operation.

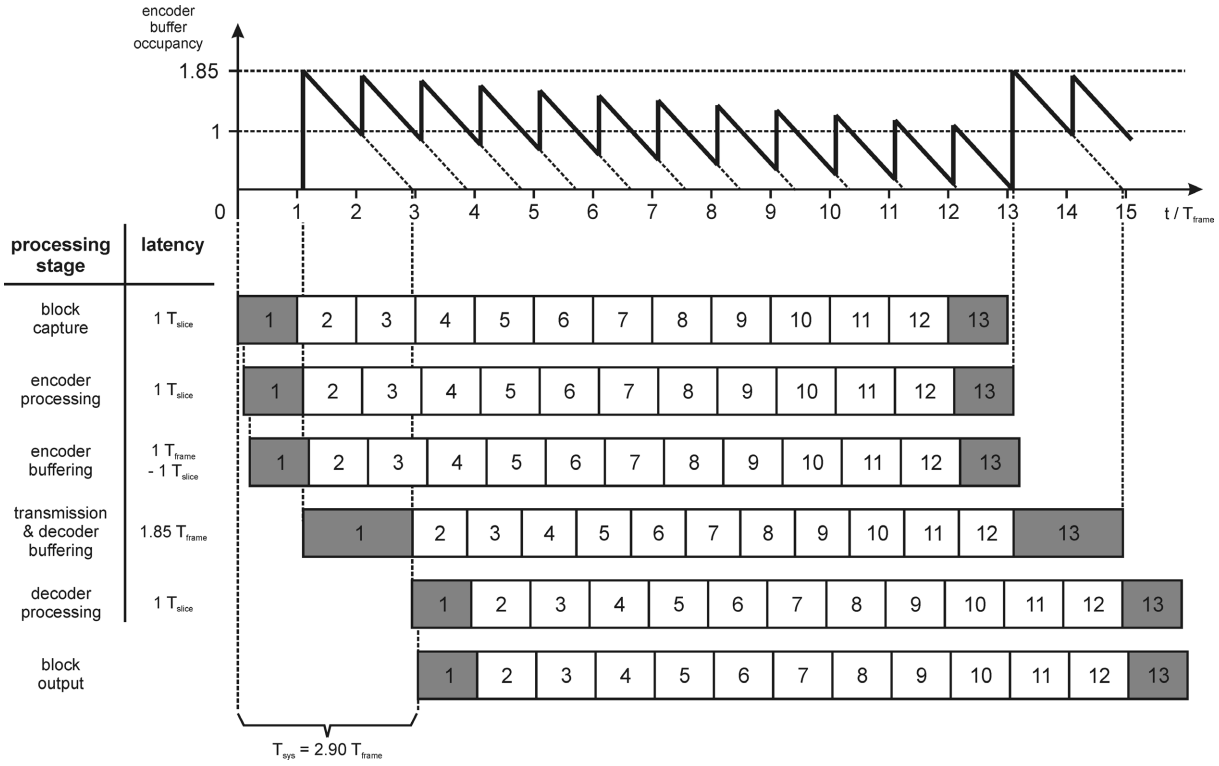


Figure 4.9: Process-scheduling for IP-coding with frame size ratio $p_ratio = 0.5$.

In practical implementations, the system works in a constant delay mode which does not allow dynamic adjustment of buffer delays according to the predictive coding gain. Therefore, a maximum buffer delay must be defined which limits the maximum allowable p_ratio during continuous operation. If the decoder buffer delay is smaller than the GOP duration T_{GOP}, the rate control must ensure that the timing is not violated by limiting the data rate of the I-frames.

The handling of scene changes is generally more critical at the beginning of a group of pictures than at the end of a GOP. Sudden scene changes usually result in a failure of the motion estimation and the first frame of a scene is encoded in intra mode even if a P-frame is scheduled. If the scene change happens to be towards the end of a group of picture, it is possible to allocate more data rate for an unscheduled I-frame due to the lower buffer occupancy.

4.4.4 Bi-directional prediction modes

For system latency analysis of the bi-directional coding modes, the additional delay introduced by the frame-reordering (section 2.3.2) and the larger dynamic of frame sizes must be considered. It should be noted that it is theoretically possible to switch dynamically between IP and IPB coding modes. However, the constant delay operation of a real-time system requires that the lower IP mode latency is increased to the IPB mode latency.

figure 4.10 shows the timed encoder and decoder frame-reordering for a typical IBBP picture structure. On the encoder side it is generally possible to encode the I and P frames as soon as they are available in the capture buffer. For encoding a B-frame, causality requires that uncompressed candidate B-frames are stored in the capture buffer until the future reference frame is available and reconstructed in the reference frame buffer. For the illustrated IBBP coding mode, the encoder introduces a delay of three frame periods for the B-frames.

The decoder can decode and display B-frames immediately, but it will delay the displaying of I and P-frames by three frame periods after the startup phase. At the beginning of a stream, the first I-frame causes an irregularity in the timing which can be neglected in the following latency analysis.

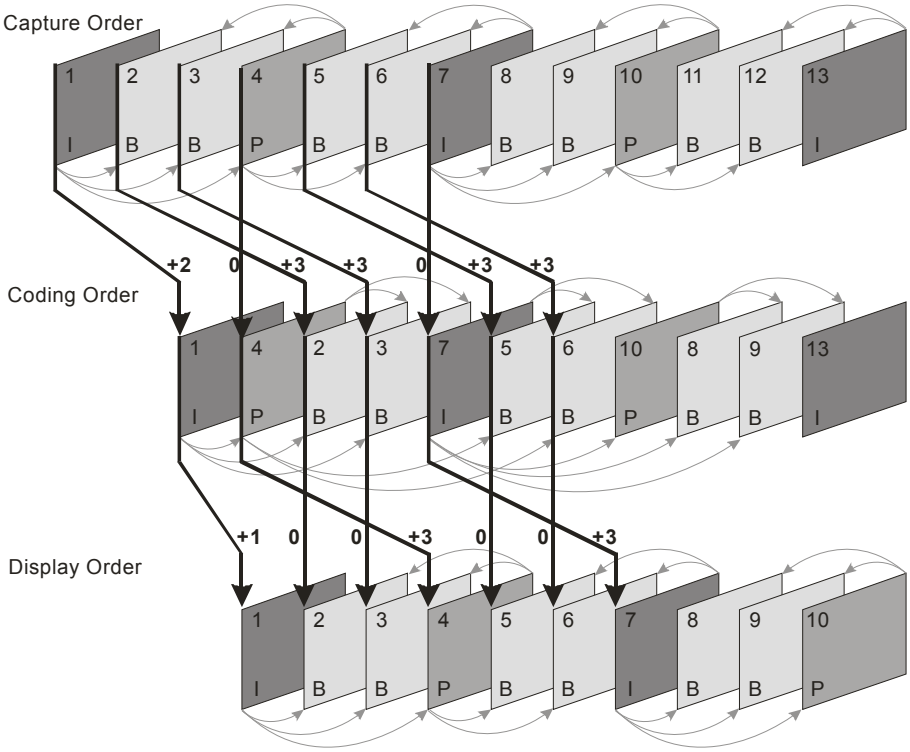


Figure 4.10: Frame reordering for IBBP coding scheme considering causality leads to three frame periods reordering latency.

From the illustration can be concluded that the overall frame reordering latency for IPB coding modes is

$$\begin{aligned} D_{reorder} &= (1+N_B) \bullet T_{frame} \quad , N_B > 0 \\ D_{reorder} &= 0 \quad , N_B = 0 \, , \end{aligned} \tag{4.7}$$

where N_B is the constant number of B-frames between the I or P-frames. In a continuous stream the uncompressed B frames are delayed in the encoder frame buffer before compression, whereas the I and P-frames are delayed in the decoder pixel buffer before displaying.

Similar to the IP-coding mode, the buffer latency of bi-directional coding modes can be calculated by

$$D_{buff,IPB} = \frac{1}{1+N_P \cdot p_ratio + N_B \cdot b_ratio} \cdot N_{GOP} \cdot T_{frame} \text{, with} \tag{4.8}$$

N_{GOP} = number of frames in GOP,
N_P = number of P-frames in GOP,
N_B = number of B-frames in GOP,
p_ratio = frame size ratio P/I frames,
b_ratio = frame size ratio B/I frames.

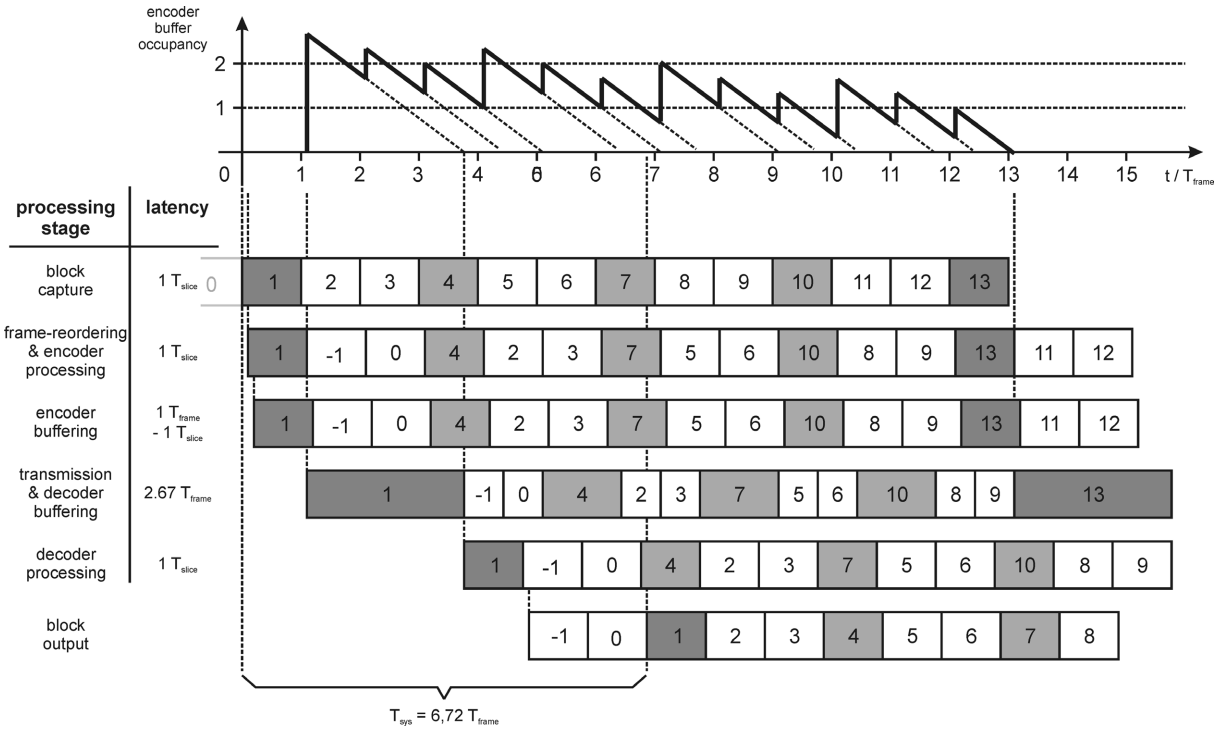


Figure 4.11: Processing and transmission scheduling for IBBP coding scheme ($p_ratio = 0.5$, $b_ratio = 0.25$, $N_p = 3$, $N_B = 8$, $N_{GOP} = 12$).

In the case of an IBBP coding mode as illustrated in figure 4.11, the decoder buffer latency evaluates to $2.67\ T_{frame}$. In the illustration the frame-reordering latency of three frame periods is located between the decoding and displaying steps for I and P-frames. Furthermore, the encoder buffer introduces a one T_{frame} delay due to the unconstrained sub-frame rate distribution. The overall system latency which satisfies all causality requirements is therefore 6.72 frame periods for slice-based processing.

By analyzing the temporal relationship between the transmission and decoding stage it can be concluded that the encoder buffer latency increases towards the end of the GOP. The IPB transmission scheme is therefore quite susceptible to scene-changes with unexpected I-frames at the beginning of the GOP.

4.4.5 Interlaced coding modes

The general scheduling concepts described in the previous sections can be applied to interlaced and progressive coding modes. For $25\ Hz$ progressive scan signals the frame period T_{frame} is equal to $40\ ms$. Interlaced video with $50\ Hz$ field scan rate reduces the frame period T_{frame} by a factor of two down to $20\ ms$.

For the special case of interlaced frame pictures (section 2.3.6) two fields of an interlaced source are combined into one frame picture prior to the encoding process. Hence, each macroblock of a frame is made up of 8 video lines of field A and 8 video lines of field B. In the generalized transmission model this effect can be considered in the block capture and block output devices as shown in figure 4.12.

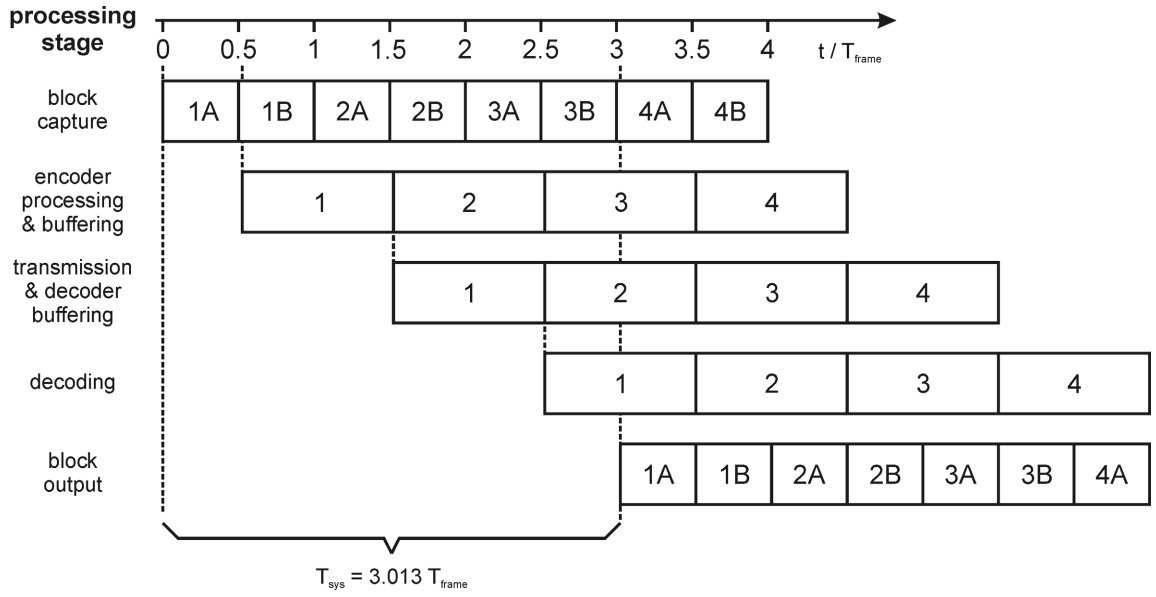


Figure 4.12: Task scheduling for intracoded frame picture transmission with interlaced video source signal.

The block capture device will generally add a $(0.5 + 0.05/2)\ T_{frame} = 0.5125\ T_{frame}$ latency to ensure the availability of second field input data before the encoder processing starts. The additional 0.0125 frame period delay is needed to capture 8 lines of the second field.

The displaying of the decoded frame is also delayed by approximately $0.5\ T_{frame}$, because the compressed data of the complete frame must be available at the decoder by the end of the first field display period. All in all, the latency for the intracoded transmission with slice processing is increased from $2.05\ T_{frame}$ for native progressive or interlaced coding to $3.013\ T_{frame}$ for interlaced frame pictures. It can be concluded that this coding mode is generally not very suitable for low-latency applications as it introduces approximately one additional frame period delay in all coding modes.

4.4.6 System latency analysis summary

In table 4.1 the results of the system latency analysis are summarized for an implementation with optimized process scheduling. The overall system latency can be estimated from the sum of the four delay sources for frame capture (D_{cap}), encoder processing (D^{e}_{proc}), buffering (D_{buff}) and decoder processing (D^{d}_{proc}). As discussed in section 4.2, the block output latency (D_{out}) can be neglected in optimized implementations.

Concerning the capture and processing delays, the majority of coding modes can be realized with a small T_{slice} delay except for two special cases: the random intra coding mode introduces two frame periods' algorithmic delay in the capture and decoder processing units. Similarly, the IPB coding mode introduces two (IBP) or three (IBBP) frame periods' algorithmic delay for frame reordering.

The bounds for the buffer delays were derived from the observation of data rate variations on frame and sub-frame level. Based on these observations an inherent CBR time interval was defined for each coding mode as described in section 4.3. It should be noted that the buffer delays D^{e}_{buff} and D^{d}_{buff} denote strictly speaking only the initial buffer delay, but according to equation 4.4 the overall buffer delay remains constant although the amount of encoder and decoder buffering changes dynamically.

For the intra IP and IPB coding modes, the initial encoder buffer delay must be set to at least one frame period if the rate distribution within the frame is unrestricted. The randomized intra mode enables shorter buffer delays due to reduced rate variations at the cost of higher algorithmic delays.

In the case of intra and randomized intra coding modes, the initial decoder buffer delay can be set equal to the encoder buffer delay. For IP and IPB coding modes, the initial decoder buffer delay is significantly higher due to the large size of the intra-coded frame with respect to the subsequent smaller predicted frames.

From the below table it can be concluded that the intra and randomized intra coding modes have exactly the same system delay of $2.05\ T_{frame}$. This is also the lower bound for the predicted coding modes if the predicted frames have the same size as the intracoded frames. As the initial decoder buffer delay increases with higher predictive coding gain, the IP and IPB coding modes have significantly higher system latencies than the other two coding modes.

Table 4.1: Coding mode summary (analytical results).

Coding mode	CBR mode	D_{cap}	D^e_{proc}	D^e_{buff}	D^d_{buff}		D^d_{proc}
					unconstrained CBR	*constrained* CBR	
intra	frame	$1\ T_{slice}$	$1\ T_{slice}$	$1\ T_{frame} - T_{slice}$	$1\ T_{frame}$		$1\ T_{slice}$
randomized intra	block (slice)	$1\ T_{frame}$	$1\ T_{block}$	0	$1\ T_{block}$		$1\ T_{frame}$
IP	GOP	$1\ T_{slice}$	$1\ T_{slice}$	$1\ T_{frame} - T_{slice}$	$1\ T_{GOP}$	$\frac{1}{1+N_P \cdot p_ratio} \cdot T_{GOP}$	$1\ T_{slice}$
IPB	GOP	$1\ T_{slice}$	$1\ T_{slice} + (1+N_B)\ T_{frame}$	$1\ T_{frame} - T_{slice}$	$1\ T_{GOP}$	$\frac{T_{GOP}}{1+N_P \cdot p_ratio + N_B \cdot b_ratio}$	$1\ T_{slice}$
interlaced frame coding		$+0.5\ T_{frame}$					$+0.5\ T_{frame}$

The system latency analysis of the previous sections is to a certain degree unsatisfactory as it overestimates the buffer delays because of inaccurate data rate modeling. Taking into account the results of the sub-frame data rate measurements of section 3.4, a high potential can be identified for optimizing the encoder and decoder buffer sizes by introducing constraints on the rate fluctuations. In the following section a detailed buffer delay analysis with real video sequences is performed to obtain reliable buffer bounds considering sub-frame rate variations.

4.5 Experimental buffer latency analysis

Conventional rate control schemes for compressed video are designed to average-out rate variations over several frames in the encoded bit stream which generally require large buffers introducing delays of multiple frame periods in the system. In this section an experimental analysis on standard video test sequences is presented which identifies the bounds for buffer management considering the rate distribution on a sub-frame level [91,92]. It is the goal of the discussion to develop a buffering concept which allows minimum latency for the transmission with a constant bitrate (CBR) using the maximum available bandwidth.

The discussions in the previous sections only considered restrictions on the amount of data per frame which results in dirac-like modeling of the encoder buffer filling at the end of each frame. This implies an unrestricted rate distribution within a frame with a worst-case assumption that $100\,\%$ of the frame data is concentrated in the last macroblock of the image. On the other extreme, a best-case scenario could be based on the assumption of completely uniform rate distribution within a frame, which would basically eliminate the need for buffering.

The experimental analysis in this section shows that it is generally necessary to consider a non-uniform distribution of data rate within a frame, because high-complexity texture or motion can be concentrated in small regions of a frame. Due to the nature of natural video sequences, the location and significance of extreme data rate variations is unknown and unrestricted but it is possible to identify natural bounds for representative test sequences. By imposing restrictions on the rate distribution within a frame, the buffering delay can be reduced at the risk of degrading the quality of video sequences which have more extreme rate distribution. The experimental buffer analysis in this section explains the approach for a sub-frame buffer analysis and compares the delay requirements of relevant coding modes based on the set of ten video sequences.

4.5.1 Leaky bucket model

The analysis and development of video buffering strategies is closely related to the reference decoder buffer control mechanisms defined in the video standards. Buffer models for the compressed data stream are generally needed to ensure proper real-time operation of an encoder-decoder video compression system with limited transmission channel bandwidth.

The model described herein is based on the MPEG-2 “virtual buffer verifier” (VBV) [18, Annex C] and H.263 “hypothetical reference decoder” (HRD) [55, Annex B] which are similar formulations of the same problem. In the original representation both models directly monitor the buffer occupancy over time, similar to the plots presented in section 4.4. In contrast to this, the “leaky bucket model” formulation

proposed in [49,56] considers the accumulated bits over time, which allows more compact graphical display of the encoder and decoder buffers in a single plot.

The leaky bucket is a direct metaphor for the encoder output buffer: the bucket is filled instantaneously with data of a complete frame and pours data with a constant rate through a "hole" in the bottom. If the bucket runs empty, channel bandwidth is lost because no data is available for transmission. On the other side, the bucket must not overflow as this would result in a loss of data in a system with real-time constraints.

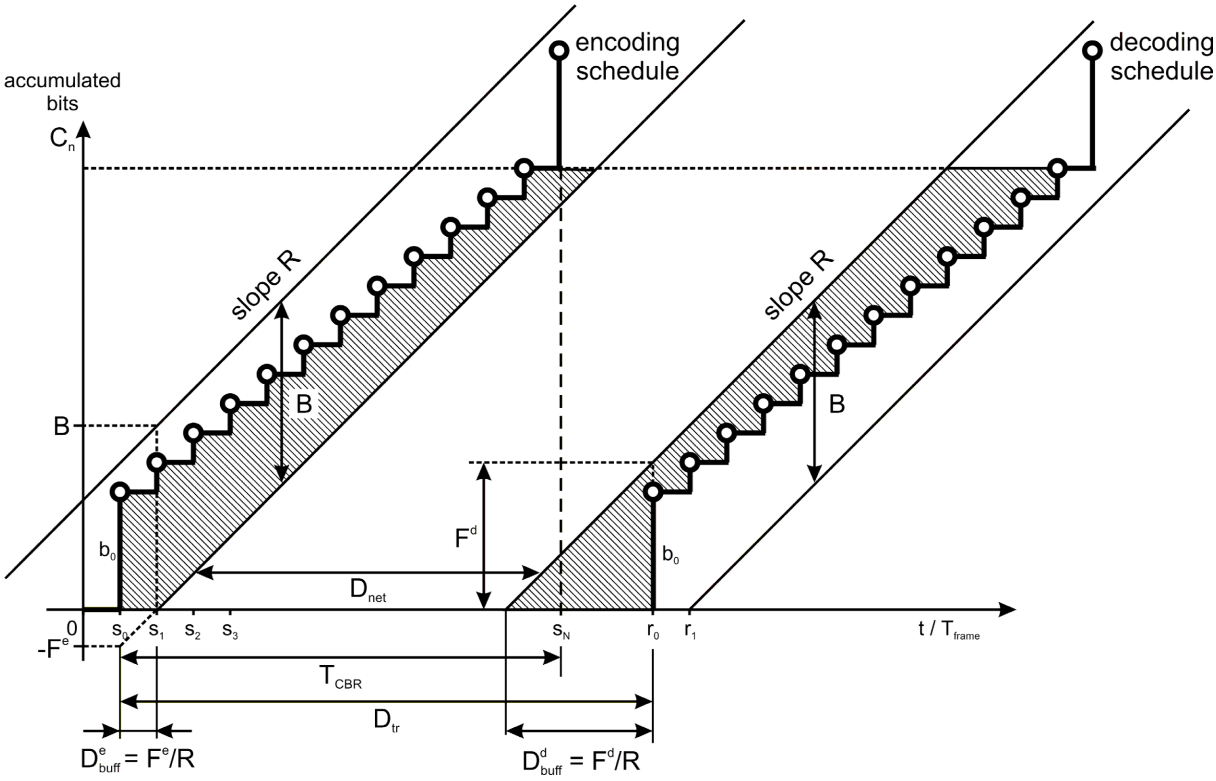


Figure 4.13: Leaky bucket model for IP-coding with GOP-CBR.

On the left hand side of figure 4.13 the encoding schedule for a video sequence is shown with frame resolution where the x-axis shows the time scale and the y-axis accumulated bits of the encoded video stream. Hence, for the ith frame of the video sequence b_i bits are placed in the encoder output buffer at discrete time instances s_i and data is drained into the transmission channel at a constant data rate R for the CBR case. In the chosen representation, the encoding schedule is represented by the staircase like number of accumulated bits

$$C_n = \sum_{i=1}^{n} b_i \,. \qquad 4.9$$

The encoder buffer has a maximum capacity of B bits. The lower bound of the encoding schedule represents the schedule on which data is poured into the transmission channel. Hence, the height of the shaded area below the encoding

schedule represents the actual occupancy of the output buffer F^e_n at frame n. Similarly, the upper bound represents the maximum capacity of the output buffer which may not be exceeded at any instance of time. Both limits are parallels with a shift of B bits.

Upon starting up the encoder, the first frame data is placed in the output buffer at time instance $s_0 = 1$ but the start of transmission is delayed by D^e_{buff}. This initial delay is related to the initial encoder buffer occupancy F^e at time s_0 by

$$D^e_{buff} = F^e / R . \tag{4.10}$$

The initial encoder buffer delay is conceptually identical to placing F^e bits of dummy data in the output buffer and starting the transmission immediately at $t = s_0$. On the decoder side, the dummy data is discarded and the decoder processing start is delayed until F^d bits of valid data are available in the input buffer. The corresponding decoding schedule is shown on the right side of figure 4.13. In the relevant CBR case the removal of a frame from the decoder buffer occurs with a constant time shift D_{tr} with respect to the encoding schedule.

Although the leaky bucket model was introduced for the encoder buffer, it also characterizes the decoder buffer as it inherits the bounds of the encoding schedule. If the decoder buffer has at least the same size B as the encoder buffer, the decoding schedule can be guaranteed by delaying the decoding process after the first bit enters the buffer according to

$$D^d_{buff} = F^d / R , \tag{4.11}$$

where F^d represents the initial decoder buffer occupancy and

$$F^e + F^d = B \tag{4.12}$$

is always satisfied. Hence, the overall buffer delay D_{buff} is equal to the buffer size divided by the data rate:

$$D_{buff} = D^e_{buff} + D^d_{buff} = B / R . \tag{4.13}$$

In the case of the decoding schedule, the height of the shaded area above the staircase indicates the actual buffer occupancy F^d_n. An under-run of the decoder buffer occurs if the decoding schedule exceeds the upper bound which corresponds to an encoder buffer over-run. Similarly, the lower bound corresponds to a decoder buffer over-run.

Finally, the end-to-end transmission delay D_{tr} can be characterized completely by the network transmission delay D_{net}, the buffer size B and the data rate R:

$$D_{tr} = D_{net} + (F^e + F^d) / R = D_{net} + B / R , \tag{4.14}$$

If the network transmission delay is neglected, the lower bound of the end-to-end delay is only defined by the ratio of the buffer size and data rate. As the minimal buffer size B is determined by the width of the encoding schedule, it is essential to reduce variations of the buffer occupancy for low delay systems.

Definition of normalized buffer parameters

Instead of using absolute data rates and bits, the relevant parameters and simulation results can be normalized to the CBR interval T_{CBR}. This allows a unified buffer analysis of different coding modes in the same simulation environment. Hence, by applying the same analysis methods for frame CBR and GOP-CBR coding methods, a valid comparison of the buffer delays becomes feasible.

In the following discussions the basic analysis block is a slice (slice resolution simulations) or a complete frame (frame resolution simulations). For each block n, the actual data rate b_i counted in bits is measured for the complete CBR interval comprising a total number of M slices or frames. The normalized sum of accumulated bits in the encoder output buffer can be calculated by

$$C_n^{'} = \sum_{i=0}^{n} b_i \Big/ \sum_{i=0}^{M-1} b_i, \quad n \in \{0 \ldots M-1\} \tag{4.15}$$

Furthermore, the average transmission rate R is given by the total number of bits generated in the CBR interval divided by the time of the CBR interval T_{CBR}.

$$R = \frac{\sum_{i=0}^{M-1} b_i}{T_{CBR}} \tag{4.16}$$

Assuming a constant draining of data from the encoder buffer with the average data rate R and start of transmission at $t = 0$, the normalized actual buffer occupancy $F_n^{e'}$ of the encoder buffer can be calculated by

$$F_n^{e'} = C_n^{'} - \frac{n}{M} = \frac{\sum_{i=0}^{n} b_i}{\sum_{i=0}^{M-1} b_i} - \frac{n}{M} \tag{4.17}$$

Hence, the encoder buffer delay which is needed for draining the bits into the channel can be calculated by

$$D_{buff,\,n}^{e} = F_n^{e'} \cdot \sum_{i=0}^{M-1} b_i \frac{1}{R} = F_n^{e'} \cdot \sum_{i=0}^{M-1} b_i \cdot \frac{T_{CBR}}{\sum_{i=0}^{M-1} b_i} = F_n^{e'} \cdot T_{CBR} \tag{4.18}$$

In the following discussions, the minimum buffer delay is derived directly from temporal data rate variations using the leaky bucket model. The model is however

not restricted to frame resolution, and the analysis will show that a slice resolution analysis results in smaller minimum buffer delays.

Derivation of buffer requirements from the encoding schedule

The leaky bucket model as illustrated in figure 4.13 describes the encoding and decoding schedule based on known buffer parameters R, F^e and F^d. Hence, in this representation the buffer delays are already adjusted to the maximum buffer requirements and the leaky bucket model serves as a verification method to verify that no buffer violations occur.

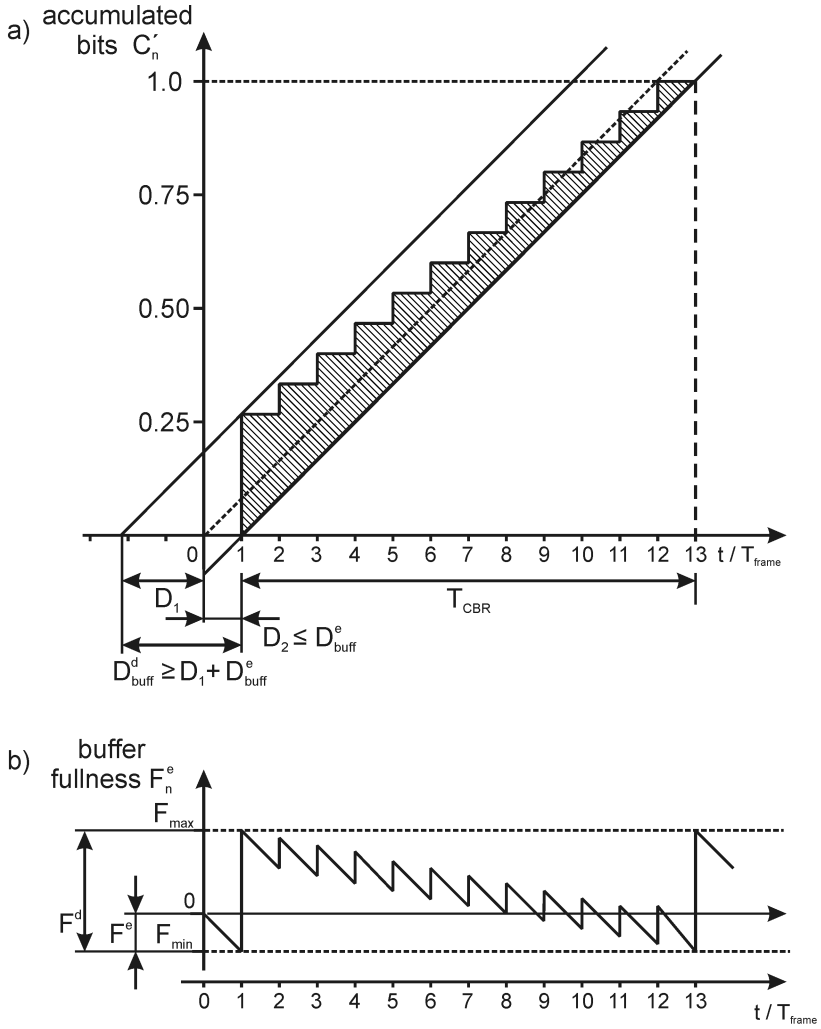


Figure 4.14: Derivation of buffer delays from the leaky bucket encoding schedule.

In contrast to this original formulation, the leaky bucket model can also be used to determine the buffer requirements from the encoding schedule as illustrated in figure 4.14. In the given example the encoding schedule (a) and the buffer occupancy (b) are shown for an IP-coded video sequence with frame-level buffer modeling. As the initial encoder delay is still unknown it is assumed that the transmission starts immediately at $t = 0$. This obviously results in an encoder buffer under-run because the first frame data is placed in the buffer at $t = 1\ T_{frame}$.

In order to avoid the encoder buffer under-run, the initial encoder buffer delay must be set larger or equal to the negative of minimum buffer occupancy divided by R:

$$D^{e}_{buff} \geq D_2 = -\min(D^{e}_{buff,n}) = -\min(F^{e'}_{n}) \cdot T_{CBR} = -F^{e}_{\min} / R \ . \quad (4.19)$$

Similarly, the initial decoder buffer delay is defined by the width of the encoding schedule which can be derived directly from the buffer occupancy:

$$D_1 = \max(D^{e}_{buff,n}) = \max(F^{e'}_{n}) \cdot T_{CBR} = F^{e}_{\max} / R \, . \quad (4.20)$$

$$D^{d}_{buff} \geq D_1 + D^{e}_{buff,n} \geq \left(\max(F^{e'}_{n}) - \min(F^{e'}_{n})\right) \cdot T_{CBR} \, . \quad (4.21)$$

From equation 4.21 it can be concluded that the initial decoder buffer D^{d}_{buff} delay is always larger than or equal to the initial encoder buffer delay D^{e}_{buff}.

As illustrated in figure 4.14 a), both buffer parameters can be derived directly from the intersection points of the upper and lower bound of the encoding schedule and the time axis (x-axis). Furthermore, if normalized buffer parameters are used, the intersection points of the encoding schedule bounds with the y-axis (C'_n) correspond to the initial buffer occupancy $F^{e'}$ and $F^{d'}$.

4.5.2 Intra coding mode

The analysis of the intra coding mode delay is based on frame-CBR operation but the buffer analysis is performed with slice resolution to achieve minimum delays. As discussed in section 4.4.1, a buffer analysis on frame resolution would result in a transmission latency of two frame periods ($D^{e}_{buff} = D^{d}_{buff} = 1 \ T_{frame}$) because all frame data is placed in the encoder buffer instantaneously after the frame is encoded completely. In contrast to this, the slice resolution buffer analysis also considers the sub-frame level data rate variations which allows significant reduction of buffer sizes and earlier start of transmission. The presented simulations are based on average macroblock data rates of the first three frames of the video sequences.

Figure 4.15 shows the buffer analysis for six selected video sequences with three simulation steps. In the first column, the distribution of data rate in the vertical dimension of the image is displayed with slice resolution. This representation is equivalent to the instantaneous temporal data rate when the slices are transmitted in sequential order. In the second column, the leaky bucket model is calculated for the encoding schedule by accumulating the instantaneous data rate of the slices in one frame. In this graph, the upper and lower bounds correspond to the minimum leaky bucket bounds.

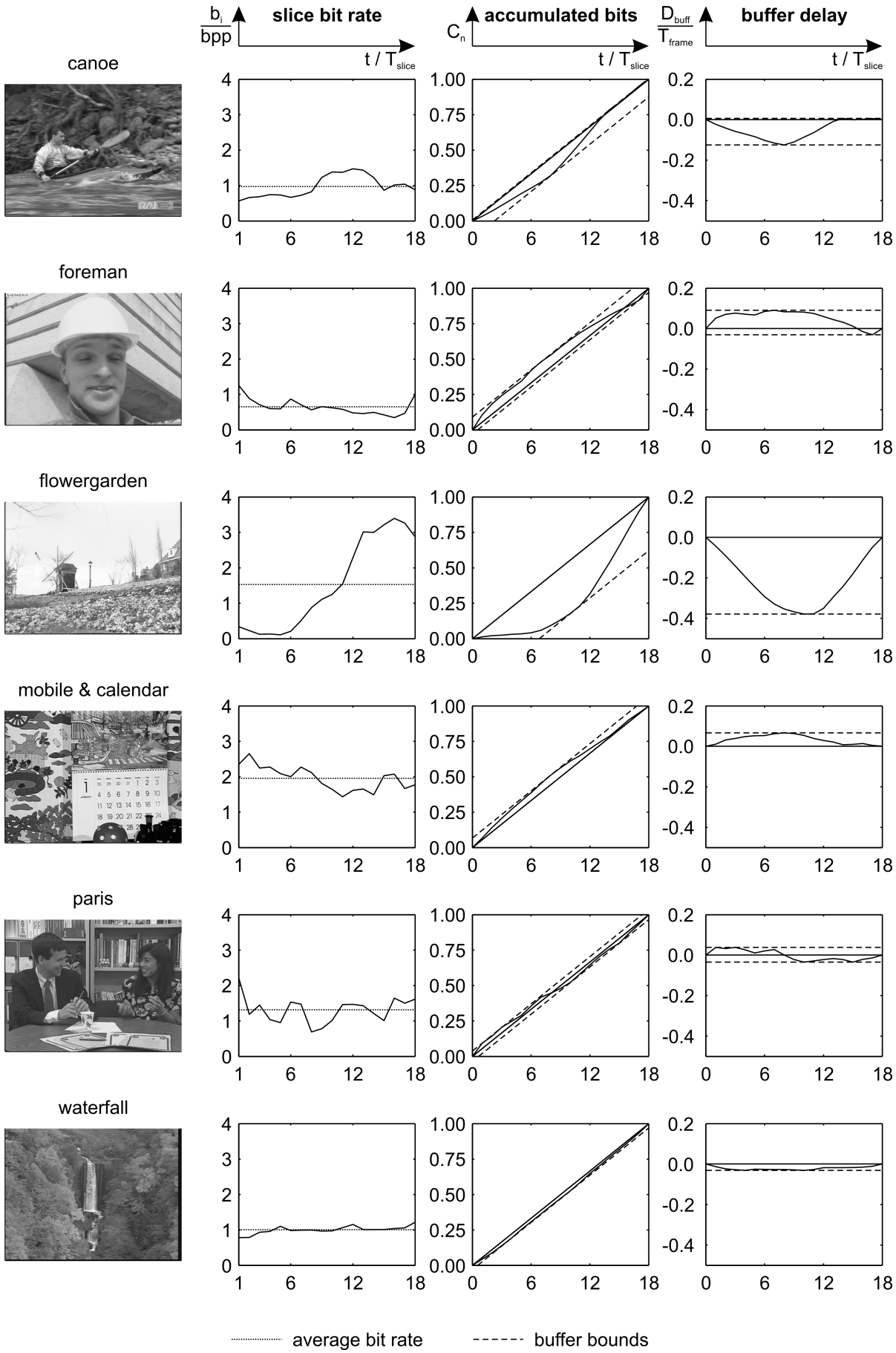


Figure 4.15: Intra coding mode buffer analysis (slice resolution).

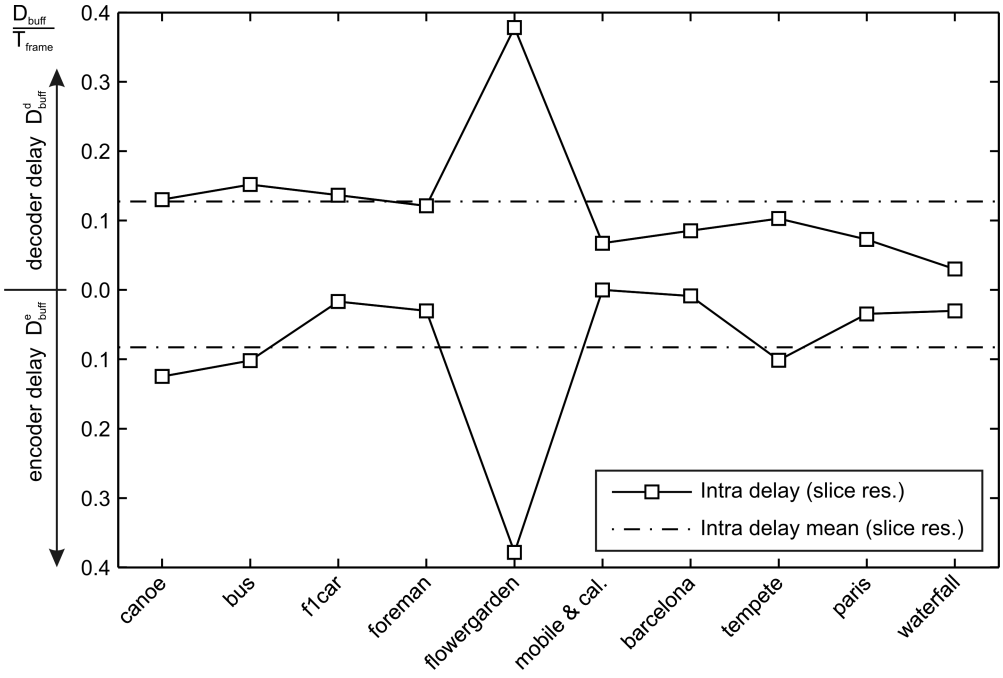


Figure 4.16: Intra coding mode delay analysis (details in table B.1).

From this intermediate step, the buffer occupancy with respect to the slice number can be derived as shown in the third column which is conceptually similar to the well-known VBV and HRD encoder buffer representation. However, in the third graph the initial buffer delay, which would be equivalent to a positive shift of the curve, is not yet introduced. Hence, the minimum of the curve represents the required encoder buffer delay in fractions of frame periods. Similarly, the sum of the maximum and the minimum value corresponds to the minimum decoder buffer delay.

The buffer analysis of the majority of video sequences shows significant deviations from the linear buffer filling which would allow shortest buffering delay. Many test sequences (e.g. *canoe*, *foreman*, *mobile & calendar*) show moderate deviations in the order of $0.1\ C'_n$. In contrast to all other sequences, the *flowergarden* test stream has a very unbalanced rate distribution with a maximum buffer occupancy of approximately $0.4\ C'_n$. This effect is caused by the low amount of texture in the upper half of the image which results in slow encoder buffer filling when the image is coded from top to bottom. Furthermore, the bottom section shows very detailed texture with a maximum slice bit rate of 3.5 bpp. In the context of low latency applications, this sequence can be considered as a worst-case scenario.

The buffer delay results from figure 4.15 are summarized and completed by four other test sequences in figure 4.16. The interpretation of this plot is identical to the third row of figure 4.15, i.e. encoder buffer delays are plotted below the zero axis and decoder buffer delays are plotted above the zero axis. With this presentation the overall buffer delay which is given by the sum of the encoder and decoder buffer delays can be derived easily. As a real-time system in continuous operation

must keep the buffer delays constant, reasonable bounds for the maximum required encoder and decoder buffer sizes must be found considering all test sequences.

It can be observed in figure 4.16 that the *flowergarden* sequence requires about three to five times higher encoder and decoder buffer delay than all other sequences. Designing a system for this worst-case scenario would therefore introduce $D_{buff} = 2 \cdot 0.38\ T_{frame} = 0.76\ T_{frame}$ buffer delay to handle similar video sequences without quality impacts. As all other test sequences would allow dramatically lower buffering delays, a tradeoff between delay adjustment and image quality of critical video sequences must be found. Setting the buffer delays to the mean values $D^{e}_{buff,mean} = 0.083\ T_{frame}$ and $D^{d}_{buff,mean} = 0.128\ T_{frame}$ would surely result in strong quality impacts in many scenarios. Based on the simulation data it can be concluded that the total buffer delay for the intra coding mode can be set in the range of 0.21 to 0.76 frame periods.

4.5.3 Predictive coding mode

Figure 4.17 shows the slice resolution buffer analysis for IP-coding with a GOP-size of 12 frames. In the first column the instantaneous slice bit rates for a single frame in intra and predicted coding mode are displayed. It should be noted that the simulations are based on the average slice data rates of the first three frames taken from the video sequence and effects of changing rate distribution over time are not considered. Although the data rate of the inter-coded slices is generally lower than the intra-coded slices, the sub frame-level rate distribution can be quite different. For example, in the *canoe* sequence the predictive coding mode data rate increases with the slice number and almost reaches the intra coding data rate in the bottom section of the frame. On the other hand, the *flowergarden* and *mobile & calendar* sequences show very similar behavior for intra-coded and predicted frames.

Applying the leaky bucket model as shown in the second column generally results in a steep encoding schedule when the initial I-frame is processed. The subsequent P-frames have a lower average data rate which results in a flat encoding schedule. For some sequences like *canoe*, *flowergarden* and *paris* a small modulation of the encoding schedule can be noticed which is caused by significant local rate variations of the predicted frames. As the I-frame is always larger than the P-frames, the encoding schedule is generally biased above the diagonal of the plot. According to the leaky bucket model it can be concluded that the decoder buffer delay is significantly larger than the encoder buffer delay. This is plausible because the I-frame data rate is higher than the average data rate which ensures that the encoder buffer is filled with sufficient data at the beginning of the GOP.

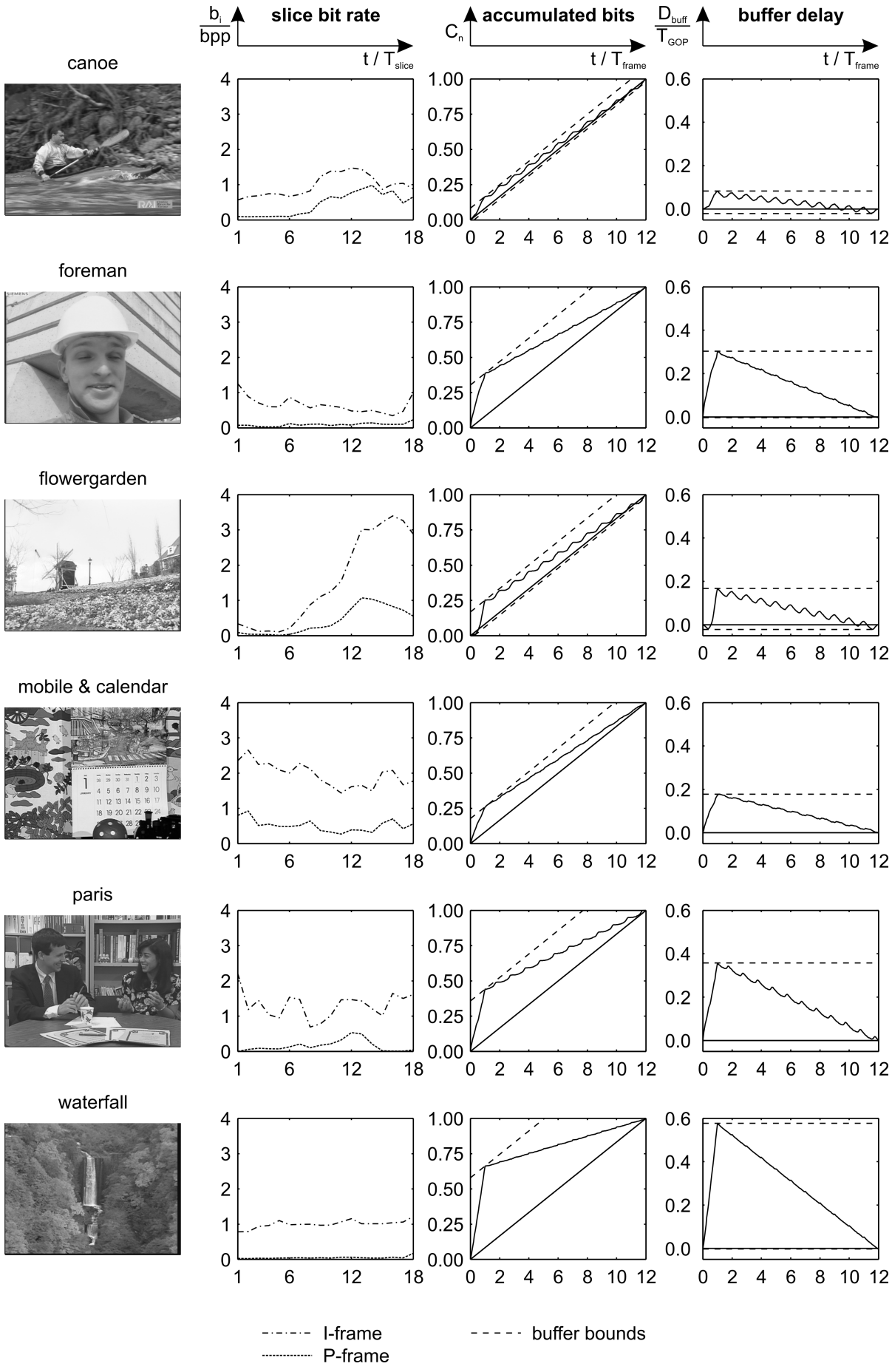


Figure 4.17: IP-mode buffer analysis (slice resolution).

It is necessary to delay the transmission start for approximately 0.25 frame periods only in cases where the initial I-frame has very low texture in the upper section (i.e. *canoe* and *flowergarden* sequence). In this way it is guaranteed that the encoder buffer does not underflow even if the transmission is started before the I-frame is encoded completely.

From the buffer occupancy analysis in the third column it can be concluded that the sequences with high predictive coding gain require the largest buffers. Especially the *waterfall* test sequence which has a $p_ratio = 0.05$ demands a very large decoder buffer delay of $0.6\ T_{GOP}$. The minimum buffer and delay requirements are further summarized in figure 4.18 for the ten selected test sequences.

As stated by equation 4.5 on page 69 the decoder buffer delay is directly related to the p_ratio displayed in figure 3.7. The five sequences with a medium p_ratio require a buffer delay of approximately 2.2 frame periods. The *canoe* sequence with the lowest predictive coding gain can be transmitted with a decoder delay of $1\ T_{frame}$ because the I-frame is only about twice as large as the P-frames. The four sequences *F1 car*, *foreman*, *paris* and *waterfall* have predictive coding gains above the average which also result in higher decoder buffer delay requirements.

The corresponding results for the MPEG-2 video standard can easily be derived from the results presented in figure 3.8. Furthermore, increasing the GOP size in any standard would generally result in higher buffer latencies according to equation 4.5.

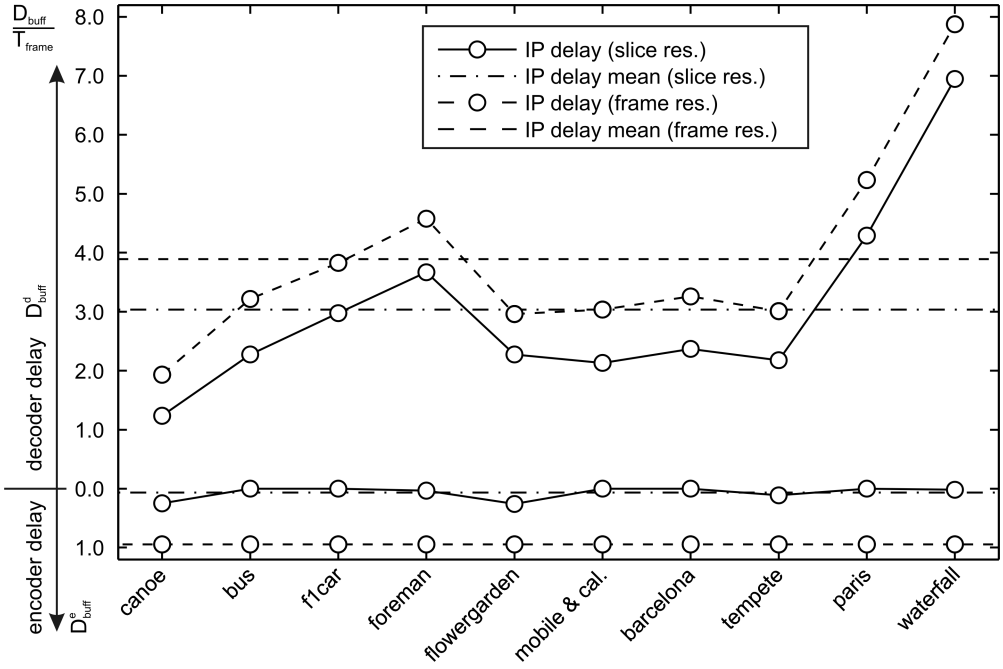


Figure 4.18: IP coding mode buffer delay analysis
(details in table B.3 and table B.4).

As the buffer and delay settings are fixed in a real-time system, a careful selection of parameters is necessary to minimize the delays in this coding mode. According to the simulations, a decoder buffer delay of 6.95 frame periods is needed for transmitting the *waterfall* sequence without quality impacts. Although seven of the sequences match the mean value of 3.0 frame periods delay, three sequences require significantly higher delays.

Limiting the delay in IP-coding mode is generally possible by limiting the size of the I-frames or increasing the data rate of P-frames with stuffing. It is also possible to combine both strategies but it is obvious that they result in unwanted quality losses. Especially in the case of H.264 where the major coding gain compared with MPEG-2 comes from the improved temporal prediction, it is an unfavorable solution to limit the predictive coding gain. The results can also be interpreted in a way that H.264 introduces higher buffer delays than MPEG-2 in order to achieve a higher compression efficiency in IP coding mode.

Concerning the encoder buffer delay, figure 4.18 proves that the slice resolution analysis can reduce the delay by 0.75 frame periods. In the case of a frame resolution analysis, all frame data is placed instantaneously in the output buffer at the end of the frame which introduces a $1.0\ T_{frame}$ delay. In contrast to this, the slice resolution simulation shows that an encoder delay of $0.26\ T_{frame}$ is sufficient even for critical video sequences like *flowergarden* and *canoe*. The increased decoder buffer delay in the frame resolution simulation is caused by the additional encoder buffer delay as indicated by equation 4.21.

4.5.4 Bi-directional prediction modes

The buffer latency of the IPB prediction modes is very similar to the IP coding mode because the I-frame consumes approximately the same proportion of data in both coding modes. As the bi-directional prediction requires additional frame-reordering, this coding mode is generally not suitable for low latency applications. Therefore, this section only gives an estimate of the buffer latencies based on frame-level simulation data of section 3.3.4 and figure 3.10. The buffer latency is estimated using equation 4.8 and the measured p_ratio and b_ratio values.

The buffer requirements for the ten test sequences in IBP and IBBP coding mode are summarized in figure 4.19 and the IP coding mode simulation result is given as reference. From the estimates it can be concluded that the decoder buffer latency for IBP coding modes is on the average $0.4\ T_{frame}$ higher than for the IP coding modes. Especially for the sequences *mobile & calendar*, *barcelona* and *tempete* the significant bi-directional coding gain results in a 0.6 to $0.8\ T_{frame}$ higher buffer delay.

Also the *foreman* sequence shows a significant increase of buffer delay although it has only a minor increased bi-directional coding gain.

The most critical test sequences *paris* and *waterfall* show only a minor increase of the buffer latency because they only have small bi-directional coding gains. It is therefore possible that the IPB coding mode has the strongest effects on test sequences with medium IP-coding gains because for test sequences with very high forward temporal prediction gains the additional gain of bi-directional prediction becomes insignificant. However, the number of test sequences is too small to generalize this observation.

Concerning the buffer delays for slice-level resolution it is very likely that the encoder buffer delay would decrease to 0.25 frame periods delay as shown for the IP coding mode in figure 4.18. Similarly, the decoder buffer delay would decrease by about $0.75\ T_{frame}$.

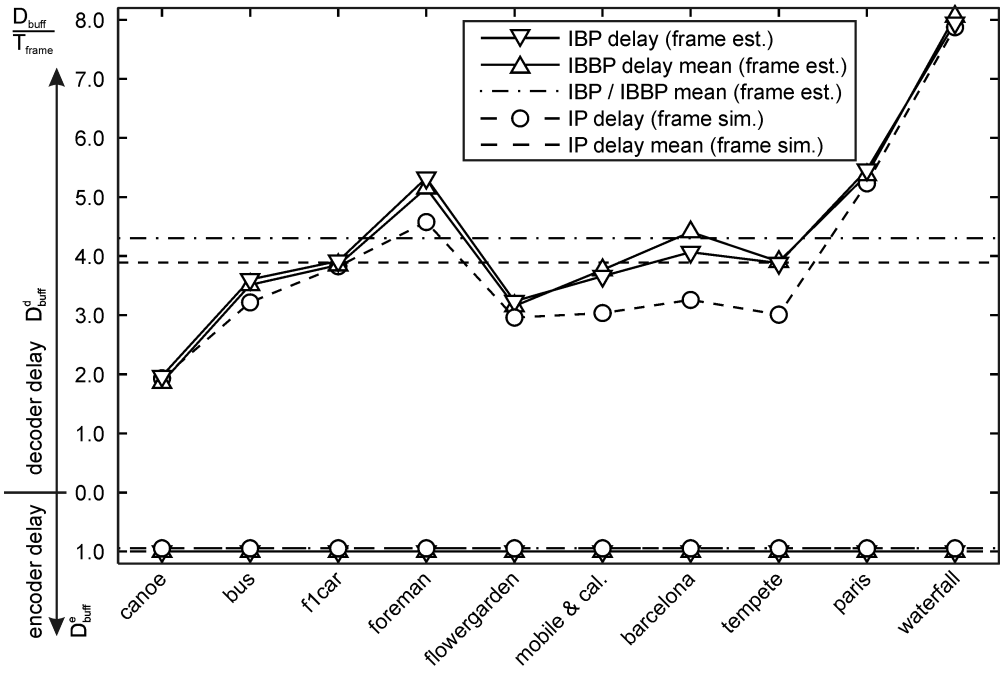


Figure 4.19: IPB coding mode buffer delay analysis
(details in table B.5 and table B.6).

4.6 System latency comparison for standard coding modes

In this section the results of the buffer and system latency analysis for the standard coding modes are summarized. While figure 4.20 only shows the encoder and decoder buffer latencies, table 4.2 includes the additional system latencies caused by algorithmic limitations.

According to the simulation results shown in figure 4.20, the IP and IPB coding modes show only small differences in buffer latencies for frame resolution simulations. For all three coding modes the decoder buffer latency must be set

significantly above four frame periods (160 *ms* for 25 *Hz* SDTV) for enabling useful predictive coding gains. The observed maximum buffer delay of 9 frame periods is however significantly below the worst-case buffer delay of 12 frames for unconstrained CBR operation. In the case of IBP and IBBP coding modes the frame reordering introduces 2 (IBP) or 3 (IBBP) frame periods algorithmic latency compared with the IP coding mode.

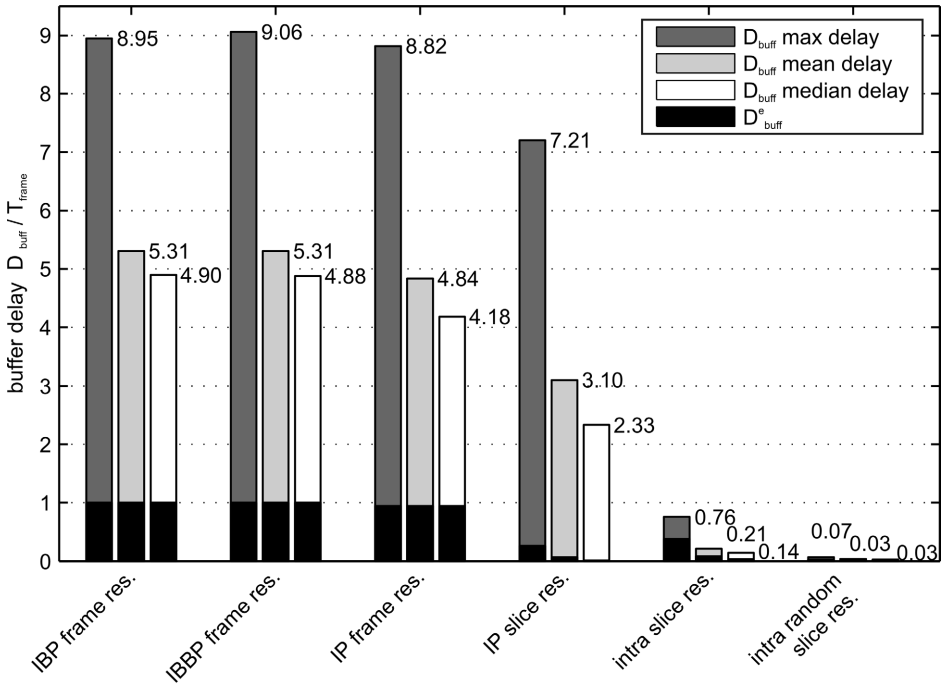


Figure 4.20: Standard coding mode buffer delay summary ($N_{GOP} = 12$). The black bars indicate the contribution of the encoder buffer delay to the total buffer delay.

On the example of the IP coding mode it can be seen that a slice-level buffer simulation which considers sub-frame data rate variations can result in a significant reduction of buffer latencies. The decrease by $1.6\ T_{frame}$ overall buffer latency is caused by the earlier start of transmission which reduces both, the encoder and decoder buffer latency by approximately 0.8 frame periods. It should be noted that the early transmission start imposes additional constraints on the rate distribution similar to the early start of decoding in constrained CBR operation.

Extremely low system delays below 1.0 frame periods with standard coding modes can only be achieved with intra coding. For the test set of 10 video sequences, the maximum buffer delays were observed for the *flowergarden* sequence. Due to the uneven distribution of texture information, the start of transmission as well as the start of decoding must be delayed by $0.38\ T_{frame}$ resulting in $0.76\ T_{frame}$ buffer latency. All other test sequences would allow significantly lower buffer delays below $0.3\ T_{frame}$ (including encoder and decoder buffers).

While the randomized intra mode offers the lowest possible buffer delays of less than 0.1 frame periods it does not allow extremely low system delays. This is due to the additional algorithmic delays of one frame period which are introduced in the encoder and decoder for the macroblock randomization process. Nevertheless, the simulation shows that coding modes with small temporal rate fluctuations can achieve significantly lower buffer latencies than standard coding modes.

For all the mentioned compression modes, the interlaced frame coding introduces an additional system delay of one frame period. Hence, if the two fields of an interlaced picture are combined into one frame prior to the encoding, the system delay is increased by 40 *ms* for 50 *Hz* interlaced source signals.

Table 4.2: Coding mode summary (experimental results).

Coding mode	**CBR mode**	D_{cap}	D^e_{proc}	$D_{buff} = D^e_{buff} + D^d_{buff}$		D^d_{proc}
				unconstrained CBR	***constrained CBR***	
Intra	frame	1 T_{slice}	1 T_{slice}	2 T_{frame}	0.21 .. 0.76 T_{frame}	1 T_{slice}
randomized Intra	block (slice)	1 T_{frame}	1 T_{slice}	2 T_{block}	0.02 .. 0.05 T_{frame}	1 T_{frame}
IP ($N_{GOP} = 12$)	GOP	1 T_{slice}	1 T_{slice}	1 T_{frame} + 1 T_{GOP}	3.10 .. 7.21 T_{frame}	1 T_{slice}
IPB ($N_{GOP} = 12$)	GOP	1 T_{slice}	1 T_{slice} + (1+N_B) T_{frame}	1 T_{frame} + 1 T_{GOP}	IBP: 5.31 .. 8.95 T_{frame} IBBP: 5.31 .. 9.06 T_{frame}	1 T_{slice}
interlaced frame coding		+0.5 T_{frame}				+0.5 T_{frame}

5 Optimized Intra Refresh Algorithm

5.1 Introduction to the intra refresh principle

The general latency analysis in chapter 4 showed that transmission latencies below $100\ ms$ cannot be achieved with standard predictive IP and IPB coding schemes, since the initial I frame in the GOP introduces a large buffering delay. In order to reduce the buffering delay, it is desirable to achieve a relatively constant number of bits per frame while maintaining the high coding gain of temporal prediction.

A very general solution to reduce the latency in compressed video transmission is the intra refresh method, where sections of predicted frames are transmitted in intra mode. Hence, instead of transmitting complete I-frames for error cancellation, the frame is intra-refreshed sequentially with smaller frame-size variations.

Apart from low-latency applications like video telephony, the intra refresh also allows flexible adaptation of error recovery performance to video content and channel characteristics of mobile receivers. Therefore, a second driving force in this field of research comes from low-bitrate coding and wireless applications where the amount of expensive intra macroblocks is minimized while a certain error robustness is maintained by strategic placement of the intra refreshed frame regions.

Intra refresh methods using a feedback channel

Previous work in the field of intra refresh video transmission can be divided into two major groups, namely systems using a feedback channel and systems without feedback [57]. If a feedback channel is available, the receiver can signal lost data to the encoder and initiate an intra update of the corrupted image regions. This technique is known as automatic retransmission request (ARQ) or selected recovery (SR). For broadcast applications without a feedback channel, only the forced intra update (FIU) can be applied which will be described later in this section.

An early implementation of an ARQ system presented in [58] initiates the transmission of an I-frame in case of a serious transmission error occurred. However, the conditional transmission of I-frames can cause mismatch errors and buffering problems as the complete I-frame transmission results in a data stream with a highly variable bit rate. A more sophisticated ARQ solution is presented in [59,60]. In this approach the error robustness is increased by additional interleaving of buffer data and a conditional retransmission of damaged data considering the error concealment of the decoder. In [61] a system is presented which tries to avoid explicit retransmission of corrupted frame data by signaling

the exact position of the corrupted transmission block back to the encoder. Based on this information, the encoder can mark the corrupted frame regions as damaged and restrict the prediction in the following P-frames to regions which are not damaged.

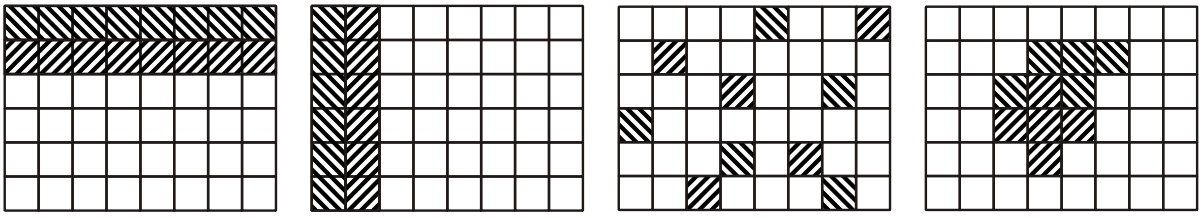
ARQ systems can generally achieve better compression since the additional intra-coded redundancy information is only transmitted when an error has occurred. On the other hand, the FIU methods can minimize the effects of transmission errors by analyzing the video scene and by inserting additional protection for important regions without the need of a feedback channel. For the problem statement of this thesis, the ARQ methods are not further considered as the desired solution should be compatible with broadcast applications.

Region-based intra refresh

In the literature three major variants of forced intra update mechanisms are described, namely region-based intra refresh, dispersed or spare intra refresh and adaptive intra refresh.

Already in 1991 a system using intracoded rows of macroblocks was proposed [62] to ease the buffer management and rate control design of hybrid video encoders. The underlying idea is to split the video frame into N_R equal-sized connected regions and intra-code all macroblocks of one region in every P-frame as illustrated in figure 5.1 a). This procedure results in a complete intra refresh after N_{GOP} frames with theoretically no overhead compared with standard IP-coding schemes with (N_{GOP} -1) P-frames per GOP.

By transmitting columns of intra-coded macroblocks rather than rows as illustrated in figure 5.1 b), the buffer occupancy variations can be further reduced resulting in a lower buffering delay [63]. The disadvantage of row-wise intra refresh lies in the fact that the macroblocks are coded in a scan-line manner which results in a high peak data rate when a long sequence (e.g. a whole slice) of intra macroblocks is encoded. In the case of row intra refresh, the temporal rate variations are smoother as only short sequences of intra macroblocks occur.



a) row b) column c) dispersed d) adaptive

Figure 5.1: Forced intra refresh strategies. Different hatches illustrate progression of intra refresh in successive frames.

For ensuring perfect recovery within one refresh cycle, it is essential that errors from previous frames cannot propagate into the refreshed regions through temporal prediction mechanisms. This can be achieved by restricting the search range of predicted macroblocks belonging to refreshed regions to refreshed regions of previous frames [62]. Throughout the document, this method of protecting the refreshed regions is referenced as *"border protection"*.

An alternative method for protecting the clean regions is proposed by [64]. Instead of restricting the motion-vector search range to "safe" macroblocks in previous frames, the intra-refresh groups generally overlap each other by at least the motion-vector search range. As the proposed method generally leads to a decreased coding efficiency due to the increased number of intra macroblocks, this approach will not be further examined.

Dispersed intra refresh

In contrast to the region based intra refresh, the dispersed refresh uses spatially isolated intra macroblocks rather than connected regions. Unlike the region based methods, it is however not possible to protect the updated regions in a dispersed refresh. Hence, the dispersed intra refresh methods generally reduce the error propagation effects by statistical means but they cannot guarantee a deterministic complete image recovery.

A dispersed refresh with random locations and a low amount of intra macroblocks per frame can be used for drift correction of the decoder in data streams which use only predicted frames. The H.263 standard [55] for example requires an intra update of a macroblock when it was transmitted 131 times in predictive mode which can be realized with a dispersed random refresh. The H.264 reference encoder [21] also includes a pseudo-random intra refresh update mechanism although the drift correction is not required by the standard as the integer transformation avoids drift problems.

In the presence of transmission errors, the percentage of intra-coded macroblocks must be increased to achieve an acceptable image quality. In [65] the propagation effects of transmission errors are analyzed analytically and a model is derived which can determine the optimal percentage of intra-coded macroblocks. The online calculation of the model parameters is described in [66], enabling dynamic adjustment of the intra macroblock refresh frequency according to the channel characteristics and properties of the video sequence.

In [67], a dispersed intra refresh method is proposed with several distinct strategies which is supposed to be less susceptible to motion compensation effects than the previously described column and row based intra refresh methods without border protection.

Although the dispersed refresh can be easily implemented it suffers significantly from corruption of the recently refreshed regions. The effect is well discussed in [68], where the error correcting performance of a random refresh with isolated intra macroblocks is compared with the performance of line or square region intra refresh in the presence of transmission errors. The analysis shows that a clustered refresh with square regions generally outperforms linear regions and dispersed intra refresh strategies. The concept of minimizing the borders of connected intra regions is further examined in [69] in the context of pseudo random location updating.

Although the dispersed refresh does not achieve the error robustness of connected regions, is has the advantage that local rate variations in the frame can be handled more effectively. Considering the measurements of localized data rates in section 3.4 it can be concluded that the dispersed intra refresh results in smaller variations of the number of bits per frame than the group or slice-based intra refresh which allows lower buffering delays.

Adaptive intra refresh

The adaptive intra refresh (AIR) methods try to minimize the effects of transmission errors by introducing cost metrics for the error probability or effects of error propagation of video frame segments. Early work in this field dates back to 1996 [70,71]. An error sensitivity metric is introduced for estimating the vulnerability for each macroblock in a frame which is based on the probability that the macroblock could not be decoded correctly in previous frames due to random channel errors. The macroblocks with the highest accumulated error probability are updated in intra mode.

Instead of cosidering the error probability of macroblocks from previous frames, [72] suggests intra-updating the macroblocks which have the highest impact on later frames by analyzing the temporal dependencies of macroblocks in consecutive frames. As the dependency graph is based on motion information of later frames, it requires an anti-causal offline operation which cannot be performed in real-time video transmission systems.

In the MPEG-4 standard [73, Annex E] an adaptive intra refresh method is described which is based on the assumption that image regions with high motion are more important and more error-prone than static image regions. Based on the observation that transmission errors in low-activity frame regions have a limited effect which can be effectively handled by error concealment algorithms, the proposed method focuses the intra refresh on high activity regions. This is achieved by generating a motion map based on the zero motion vector SAD of two consecutive frames. If a macroblock SAD exceeds a certain threshold, it is

considered as a high-motion region and the macroblock will be intra-refreshed in the next frame or as soon as bandwidth is available.

The algorithm proposed in [74] adds motion adaptation to the MPEG-4 AIR in a way that the percentage of intra-coded macroblocks is adapted to the overall motion activity in the video sequence. If low and medium motion activity is detected, the intra refresh rate is increased linearly with the fraction of motion vector data related to the overall data rate. In high motion video sequences, the intra refresh rate is however decreased for better visual performance, but when a change to low motion activity is detected the intra refresh is reenforced. This strategy allows fast recovery of the video frame after high-motion sequences as the visual effects of errors are often more disturbing in low-motion videos.

A similar approach to the MPEG-4 AIR is also taken in the patents [75] and [76] but instead of using a motion map, the algorithm [75] selects the macroblock with highest SAD value and its eight surrounding neigbours for the intra refresh. Respectively, [76] uses pre-defined groups of macroblocks for the decision and update. Furthermore, both references suggest using a dispersed refresh for the background area to ensure that the whole frame is eventually intra refreshed.

Summarizing the discussion it can be concluded that the AIR methods cannot guarantee perfect picture reconstruction like the region-based methods. However, the concept of adapting the intra refresh to motion information can also be adapted for the region based intra refresh. In the next sections these ideas will be combined in a way that good error performance and optimal coding efficiency can be achieved at the same time.

5.2 An adaptive region-based intra refresh scheme

5.2.1 Region-based forced intra refresh methods

From the discussion in the previous section it can be concluded that only the region-based refresh can ensure a safe and complete picture refresh within a defined worst-case time interval. In the following discussion a modified region-based intra refresh method is presented which enables rate-optimized intra refresh video transmission with predictive coding at very low delay.

Instead of encoding one large I-frame at the beginning of the group of pictures, the intracoded macroblocks are distributed over a sequence of predicted frames. As illustrated in figure 5.2, the video frame is split into N_R regions and one region is transmitted in intra mode in each frame while the others are transmitted in predicted mode. Within a predefined time interval of one refresh cycle consisting

of N_R frames, each region is transmitted once in intra mode and $N_R - 1$ times in predicted mode.

Assuming a uniform data rate distribution for intra and inter prediction modes within in a frame, transmitting the same number of intracoded macroblocks per frame will result in a constant bit rate per frame. Consequently, the CBR-interval can be reduced from a complete GOP to approximately one frame and the transmission scheduling becomes identical with the intra coding mode slice processing described in section 4.4.1. However, due to the varying amount of texture within a frame, variations of the region data rate can be observed which lead to significant variations of the frame sizes. For this reason, a buffer analysis for the frame-CBR mode as well as a GOP-CBR mode are presented in section 5.7.

As indicated in [62] and [64], the forced intra refresh method requires a protection mechanism to ensure completely error-free display after one complete refresh cycle was transmitted. The principle of the protection mechanism is illustrated in figure 5.2. At the beginning of a cycle, all regions are marked as potentially corrupted "dirty" regions, which become "clean" regions when they were intra-coded in a previous frame of the same refresh cycle.

For the encoding of the current frame, only clean regions of reference frames may be used for the temporal prediction of clean regions of the current frame. Otherwise, if prediction from a dirty region would be allowed, errors could propagate into the clean regions, thus destroying the intra refreshed regions. When the whole frame was refreshed, all regions are marked as dirty again and a new refresh cycle begins. Thus, each refresh cycle is performed independently and it is generally assumed that the frame could be corrupted at the beginning of a refresh cycle. A worst-case error-recovery can be guaranteed within $(2 \cdot N_{GOP} - 1)$ frames and, as a side-effect, the video stream becomes re-entrant at the beginning of each refresh cycle.

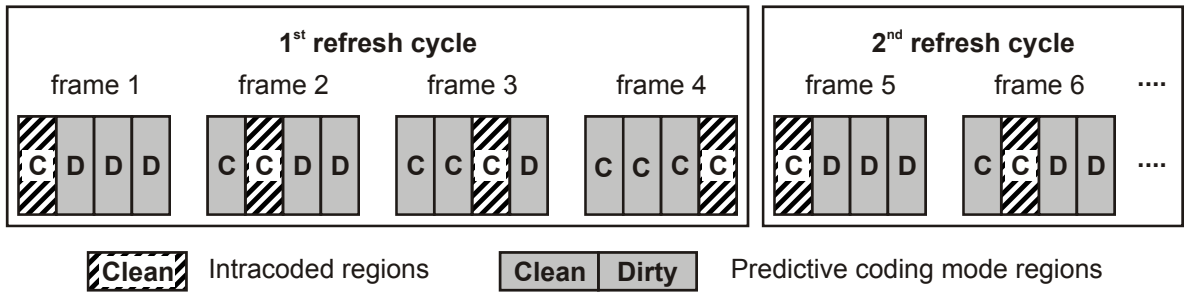


Figure 5.2: Principle of a region-based intra refresh.

It should be noted that there is not necessarily an intra-coded region between spatially adjacent clean and dirty image regions as figure 5.2 might suggest. From the algorithmic point of view it is only relevant which regions of the previous frame are marked as clean, as the currently refreshed region does not have an

effect on the motion search range restriction of the current frame. It is also possible to insert predicted frames without intracoded macroblocks into the video stream as long as the clean regions do not use reference data from dirty regions.

One drawback of the described method is that the restricted search range can lead to a decreased coding performance for the border macroblocks of clean regions as illustrated in figure 5.3. Especially in cases where the global motion is pointing from the dirty towards the clean region, the border macroblocks of clean regions would require reference data from dirty regions. As this is prohibited by the restricted search range, the best matching reference region cannot be found and the temporal prediction becomes less effective.

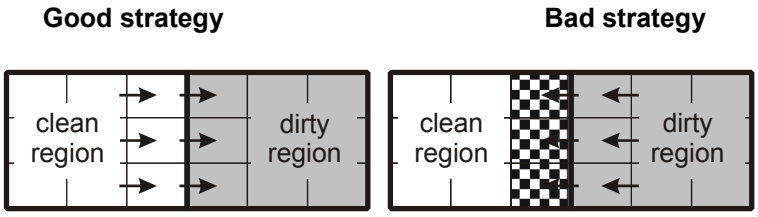


Figure 5.3: Intra-refresh border protection. The critical chequered border blocks are frequently intra-coded. Arrows represent motion vectors for temporal prediction.

Coding experiments have shown that especially for border blocks the amount of data for coding motion vectors and DCT information increases significantly due to the discontinuity of the motion vector field and higher energy of the error signal. Furthermore, significantly more intra-coded macroblocks were observed in the border regions because the encoder may choose to encode the macroblock in intra mode rather than coding a large differential error signal.

5.2.2 Motion-adaptive intra refresh method

To overcome the problem of increased data rate due to border protection, a motion-adaptive algorithm inspired by the AIR methods is proposed and evaluated in the following sections. The basic idea is to adapt the intra refresh strategy to the global motion of the sequence as illustrated in figure 5.3. An optimal strategy minimizes the costly intra-coded border blocks and avoids the increase of data rate.

For finding the optimal strategy it is generally assumed, that the increased data rate due to the border protection is exclusively caused by motion vectors pointing from dirty to clean regions. Hence, prediction within a clean or dirty region as well as prediction from clean regions into dirty regions does not affect the coding efficiency.

A fine-granular approach for this problem would be to start the refresh with a fixed number of intra-macroblocks per frame in the area which is used as reference for motion prediction most frequently, i.e. the center of the motion trajectories. Starting form the prediction center, the refresh would propagate along the direction of the motion vectors and finally end at the image borders. This method is illustrated in figure 5.4. If the motion vector field is continuous and has a clearly defined center, it should be possible to perform a safe intra refresh without border protection mechanisms.

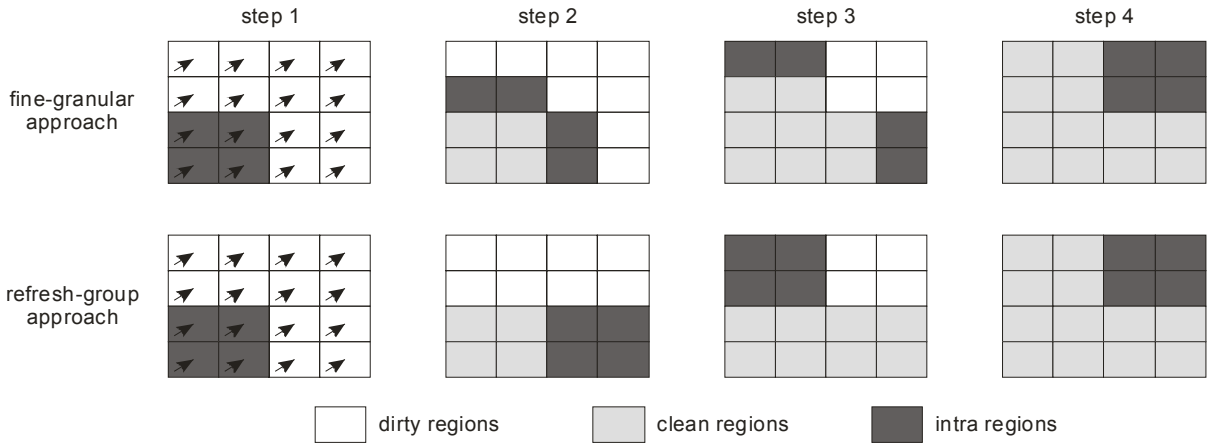


Figure 5.4: Fine-granular and refresh-group intra refresh.

The best adaptation could be achieved with a granularity of single macroblocks. However, finding the prediction center and adapting the strategy according to the motion vector field would be a computationally costly task as a large number of motion vectors must be analyzed, e.g. 1620 macroblock motion vectors in a SDTV video frame. Furthermore it should be considered that the motion may change within a refresh cycle which is especially critical if a fine-granular pattern is chosen.

To overcome this rather complex problem an adaptive refresh-group based algorithm was proposed and evaluated [88,89,90,93]. The idea is to combine sections of the frame to equally sized refresh groups and choose the optimal refresh sequence based on the average motion of the whole group. Reducing the number of groups simplifies the cost estimation and allows simple storage and recalling of different strategies.

In contrast to the row or column wise intra refresh methods proposed in [62] and [63], the frame is split into $N_R = k \cdot l = 12$ approximately square regions as illustrated in figure 5.5. By using square regions for the intra refresh, improved adaptation to complex motion of the video sequence can be achieved. Although the fine-granular approach would offer even more flexibility and adaptability for complex motion, a number of 12 refresh-groups proved sufficient for a good performance even with

difficult sequences. In order to reduce the percentage of intra-coded macroblocks it is also possible to increase the number of regions.

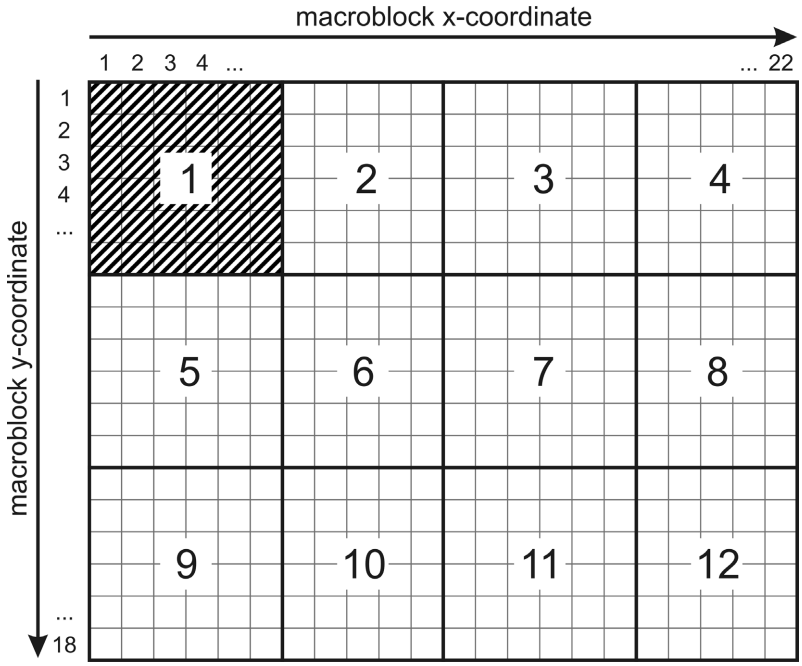


Figure 5.5: Twelve-region intra refresh macroblock grouping for CIF resolution. Macroblocks with coordinate (x,y) are assigned to regions 1 to 12 ($k = 4, l = 3$).

Theoretically, the twelve regions can be refreshed in $N_R! = 12! = 479 \cdot 10^6$ different sequences in random order. To reduce the complexity of the algorithms a set of $K = 30$ possible sequences ("refresh patterns") shown in figure 5.6 is defined which is related to frequent camera and object movements like pan, tilt and zoom. Each camera action was considered with several rotated and mirrored possibilities leading to a set of logical strategies. The refresh patterns used in the simulations may still be incomplete but for all test cases it could be shown that at least one of the strategies matched the requirements, thus leading to no significant increase in data rate compared with the standard IP-coding method.

For the definition of the predefined patterns the overall border cost between dirty and clean regions should be taken into account. As the possible effects of intra refresh border protection can be reduced by minimizing the border length between dirty and clean regions, the predefined patterns use only connected regions when proceeding through the sequence. Assuming that completely random global motion is not likely to occur in real video sequences, it is also not very likely that other random refresh sequences will outperform the predefined sequences significantly as they will result in unlinked clean regions with extra border protection cost in most cases.

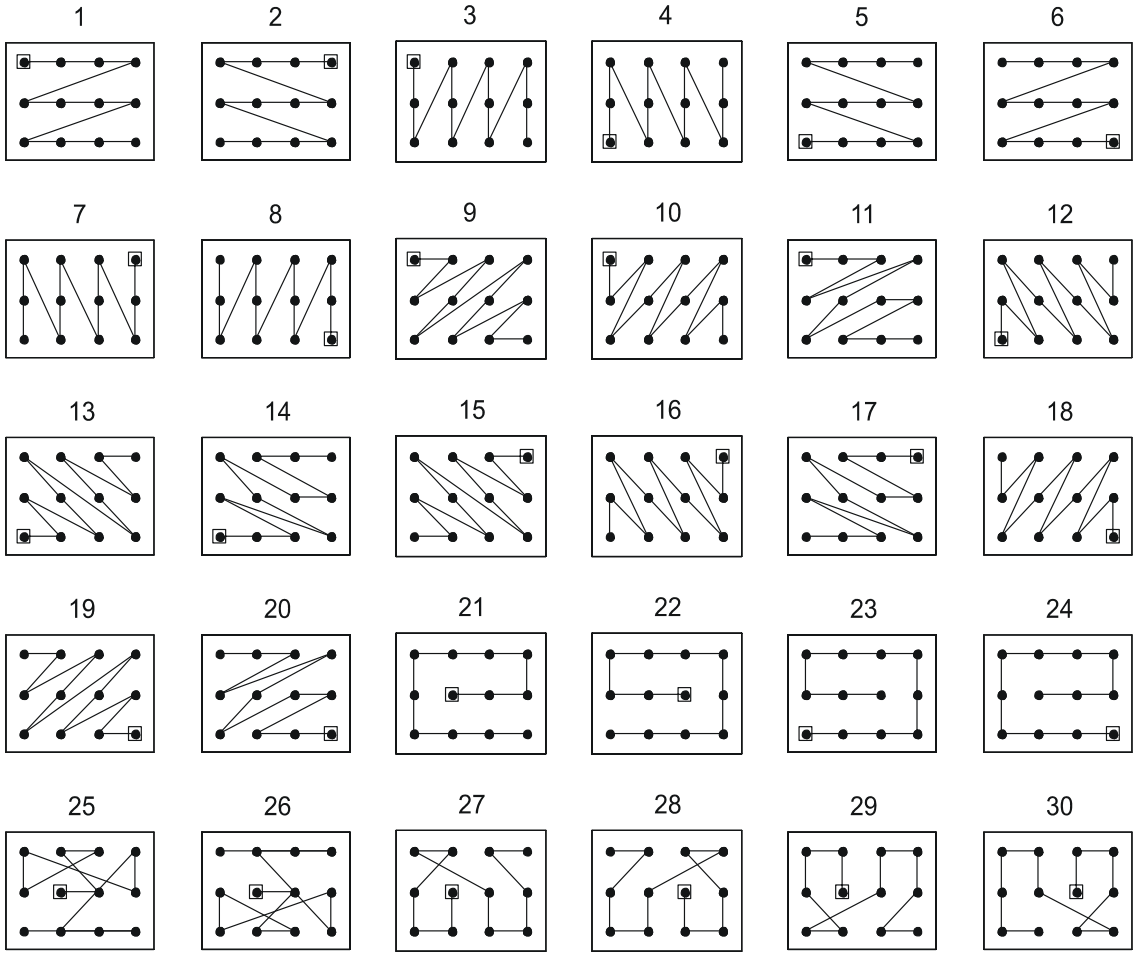


Figure 5.6: The $K = 30$ refresh patterns used in the simulations. The refresh starts at the region with the square mark and proceeds with the line.

By introducing a small set of possible refresh sequences, the problem of finding one optimal pattern for the next refresh cycle is reduced to a simple analysis of the region motion and a simulation of the border protection effects for the predefined refresh patterns. In offline video coding systems it would be possible to run a motion analysis for the complete next refresh cycle before the actual coding pass to find the optimal pattern.

As the intra refresh is mainly targeted for low-latency applications, the selection of refresh patterns is always based on motion information from previously encoded frames. The selection of an optimal refresh pattern is based on the assumption that continuity in the direction of motion exists for the duration of a complete refresh cycle. Therefore, a sudden change in the global motion direction can cause an increased number of border-protection macroblocks. As the selection of optimal patterns is only based on the relative movement of regions, this effect would only occur if the global motion is completely reversed within a refresh cycle. For the targeted applications it is however more likely that the camera movement slows down before it is reversed, which gives the intra refresh enough time to adapt.

5.2.3 Selection algorithm for optimal refresh

5.2.3.1 Proposed selection and cost calculation algorithm

The problem of finding the optimal refresh pattern for a video sequence is solved by analyzing certain characteristics of the video sequence. A cost function is calculated for every refresh pattern based on these characteristics and the refresh pattern with the smallest sum of estimated cost will be chosen for the next refresh cycle.

The measure is based on the intrusion of dirty regions into clean regions which must be avoided by using a border protection mechanism. As discussed in section 5.1, the reduced search range for temporal prediction frequently leads to a failure of the motion prediction resulting in an increase of data rate.

Before a refresh cycle is executed, the border protection cost of different strategies is estimated based on the average motion of the regions as illustrated in figure 5.7. The cost function $C_{k,n,i,j}$ between regions i and j in frame n of refresh pattern k is defined as follows:

If the current region i is "clean" and the neighboring region j is "dirty", and the average motion component (X_j, Y_j) of region j is pointing to region i:

$$C_{k,n,i,j} = \begin{cases} |X_j| & \text{for horizontally adjacent regions,} \\ |Y_j| & \text{for vertically adjacent regions.} \end{cases} \quad (5.1)$$

For all other cases:

$$C_{k,n,i,j} = 0$$

The $C_{k,n,i,j}$ costs are calculated and accumulated for all direct neighbors $j \leq 4$ of a clean region i. For each frame n all clean regions with all neighbors are considered and the cost is accumulated for all N_R frames of a refresh cycle. Finally the overall cost

$$C_k = \sum_{n=1}^{N_R} \sum_{i=1}^{N_R} \sum_{j=1}^{4} C_{k,n,i,j} \quad (5.2)$$

is compared for all K refresh strategies. The linear accounting of the motion component is motivated by the observation that the error signal energy of a border macroblock increases when its search range is strongly limited in the direction of the best matching reference frame area. A detailed flow-chart of the algorithm implementation is described in section 5.3.3.

The estimation principle is illustrated for the example in figure 5.7 with global motion two units to the left and one unit up. In frame 1, region 1 is intra-refreshed and no border protection is necessary. For frame 2, region 1 needs to be protected

against horizontal intrusion of region 2, resulting in a penalty of two. Furthermore region 1 must be protected against vertical intrusion of region 3, accounted as a penalty one. As shown in figure 5.7, the cost of all steps will accumulate to eight. The reader can easily verify, that the optimum strategies (4,3,2,1) and (4,2,3,1) will result in zero cost as expected.

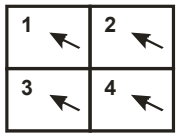
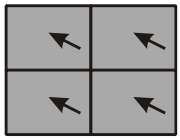
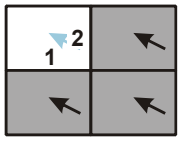
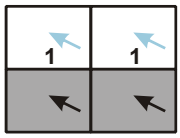
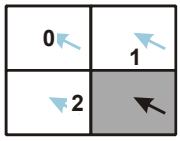
Example scenario:		refresh sequence: average motion:	(1,2,3,4) (X,Y)=(-2,-1)
frame 1		No border protection.	
frame 2		region 1: clean region 2: dirty region 3: dirty region 4: dirty	**2+1** **0** **0** **0**
frame 3		region 1: clean region 2: clean region 3: dirty region 4: dirty	**1** **1** **0** **0**
frame 4		region 1: clean region 2: clean region 3: clean region 4: dirty	**0** **1** **2** **0**
		Accumulated cost	**8**

Figure 5.7: Example cost estimation for refresh sequence (1,2,3,4) with average motion (X,Y)=(-2,-1) for all regions. Unprotected "dirty" regions are marked with gray, protected "clean" regions are white.

The average motion can be calculated for each region by averaging all motion-vectors of all macroblocks of the previous refresh cycle. As border macroblocks show some irregularities in motion prediction due to the restricted search range, they should not be considered in the calculation.

It should be noted that only the penalty for prediction of clean regions from dirty regions is considered. The other three possible cases (dirty-to-dirty prediction, clean-to-dirty prediction, clean-to-clean prediction) are not considered as they have no significant effect on the data rate. Although it is the intention of the algorithm to maximize the prediction from clean into dirty regions, considering the clean-to-dirty prediction rather than the dirty-to-clean prediction did not result in a better correlation with the measured data rates.

Simulation experiments show that it is not necessary to trace and consider motion of objects explicitly to find a good refresh pattern. As a refresh cycle usually takes 0.5 to 1.0 second it is quite likely that object movement changes significantly, thus a strong moving object-based strategy could even lead to decreased performance.

5.2.3.2 Simulation set-up

The correctness of the proposed estimation algorithm was verified with extensive simulations using the test set-up shown in figure 5.8. In contrast to the practical implementation described in section 5.3.1, the test set up uses multiple encoding runs of the same video stream for measuring the excess data rates and generating the cost estimates.

All simulations start with the initialization of the mapping memory in step 1) which groups individual macroblocks of a frame into N_R regions. In step 2) the refresh patterns are initialized as defined in appendix E. The first simulation using the program flow on the left hand side generates the measure for the reference file size. This is achieved by encoding the initial I-frame and 11 P-frames without intra refreshing the P-frames. This reference stream has the same number of intra-coded and predicted macroblocks as the intra refreshed streams except for minor effects of intra-coded macroblocks caused by failed temporal prediction. Skipping the first frame of the video sequence also ensures that the reference stream uses exactly the same frames as the intra refreshed sequences.

The second simulation illustrated in the middle analyzes the region motion of all 12 frames in the refresh cycle. The average motion is determined by calculating the mean value of the x and y motion vector components of all macroblocks in all frames belonging to the same region. Afterwards, the cost estimates for all defined refresh patterns are calculated. The estimation of intra refresh costs are based on the motion vectors of standard IP-coding of the corresponding video frames in order to avoid feedback of the intra refresh on the simulation.

For measuring the actual overhead data rate, the video sequence is encoded with all K refresh patterns and the file size of the 12 P-frames is measured. After each encoding run, the input video sequence is reset to the first frame.

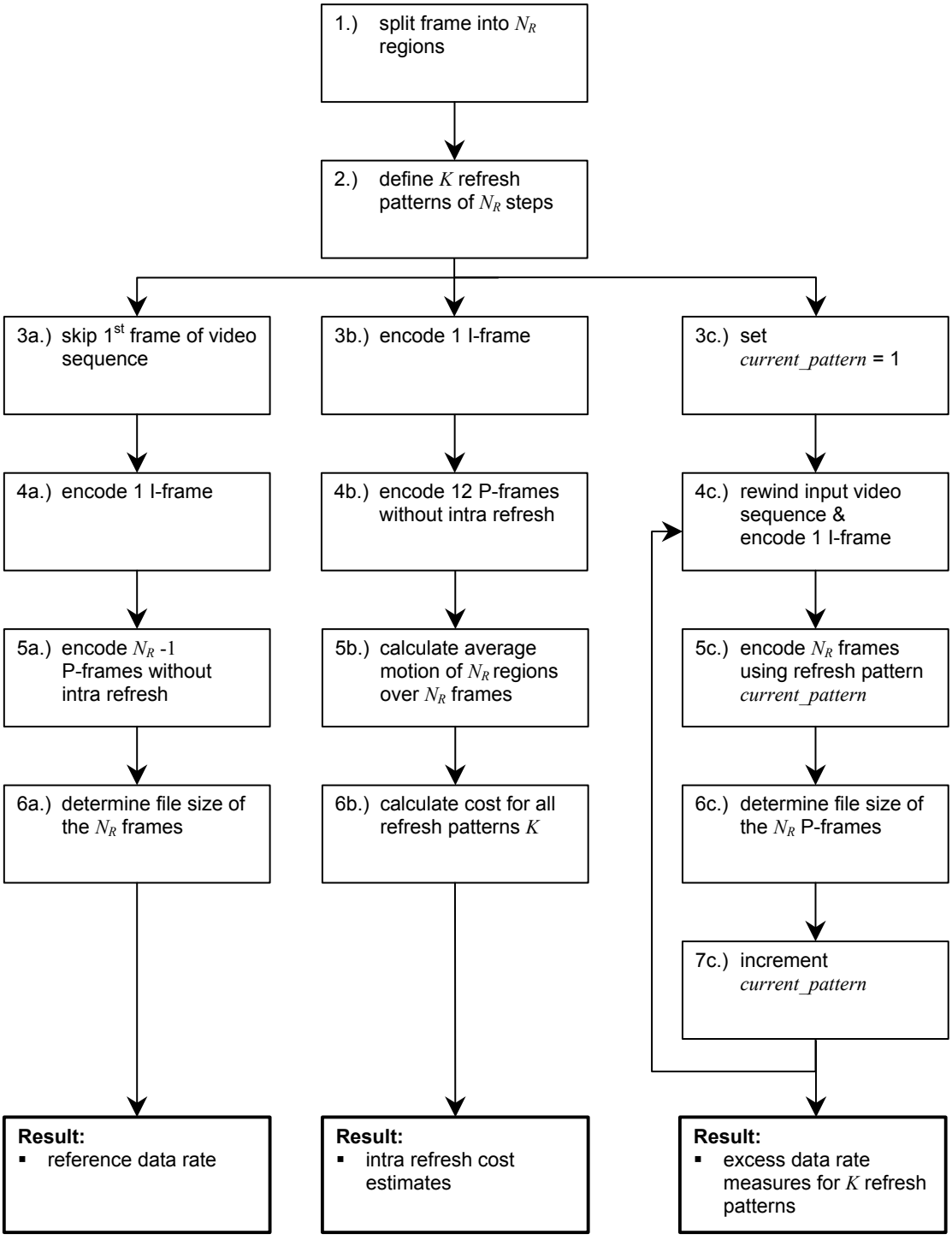


Figure 5.8: Simulation set up.

5.2.3.3 Simulation results

In this section simulation results for the H.264 JM 7.5b reference encoder [21] are presented. All tested video sequences are 25 *Hz* progressive scan CIF resolution (352 by 288 pixels). The coding parameters are set to IP-coding with one reference frame, search range +/- 16 pixels, constrained intra prediction and disabled deblocking filter (appendix F).

The simulation results in figure 5.10 are sorted by decreasing motion activity. Excess rate measures are calculated by comparing the file size of 12 compressed frames with forced intra refresh and a video stream with standard IP-coding. The size of the initial I-frame which is necessary for starting the intra refresh was subtracted. As the proposed algorithm only gives a relative rating of the refresh patterns, the estimation results were scaled to match the value range of the rate measurements. The absolute value of the excess rate estimate is not relevant and cannot be used for calculating the absolute file size. The average region motion of the video sequences is illustrated in figure 5.9.

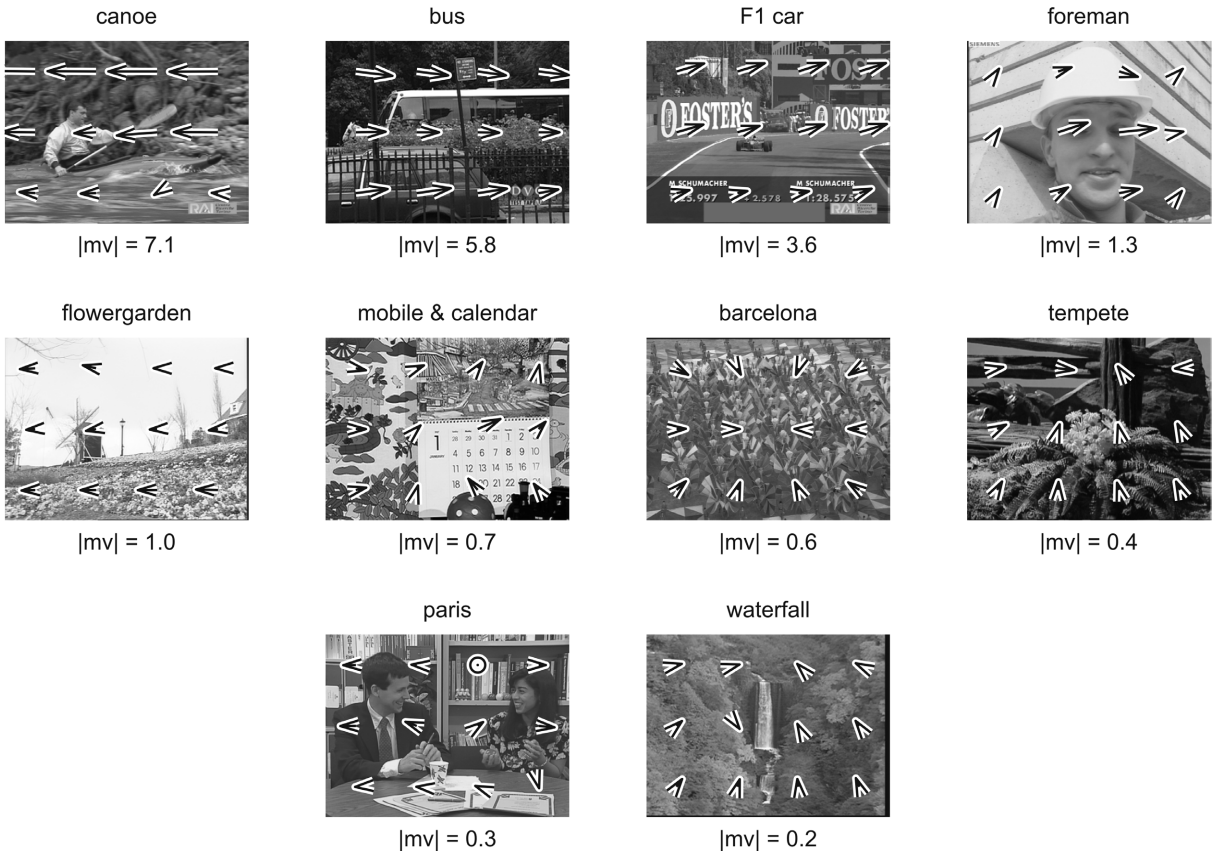


Figure 5.9: Average region motion vectors calculated by the simulation. The length of motion vectors is adaptively scaled per sequence for better visibility.

A rough analysis of the excess rate measures in figure 5.10 shows, that bad intra refresh strategies require between 5 % and 12 % excess data rate with two scenes (*bus*, *waterfall*) having more than 10 % excess data rate. In addition it can be observed that the peak excess rate does not generally increase with high motion activity but it seems to be related to the amount of detail present in the image. In many cases the optimal refresh pattern requires a slightly lower data rate than the IP-reference which may be caused by complexity variations of the intra refresh regions in later frames. Furthermore, it can be observed that for all tested video sequences several good refresh patterns exist with only 1 % or 2 % loss in coding efficiency.

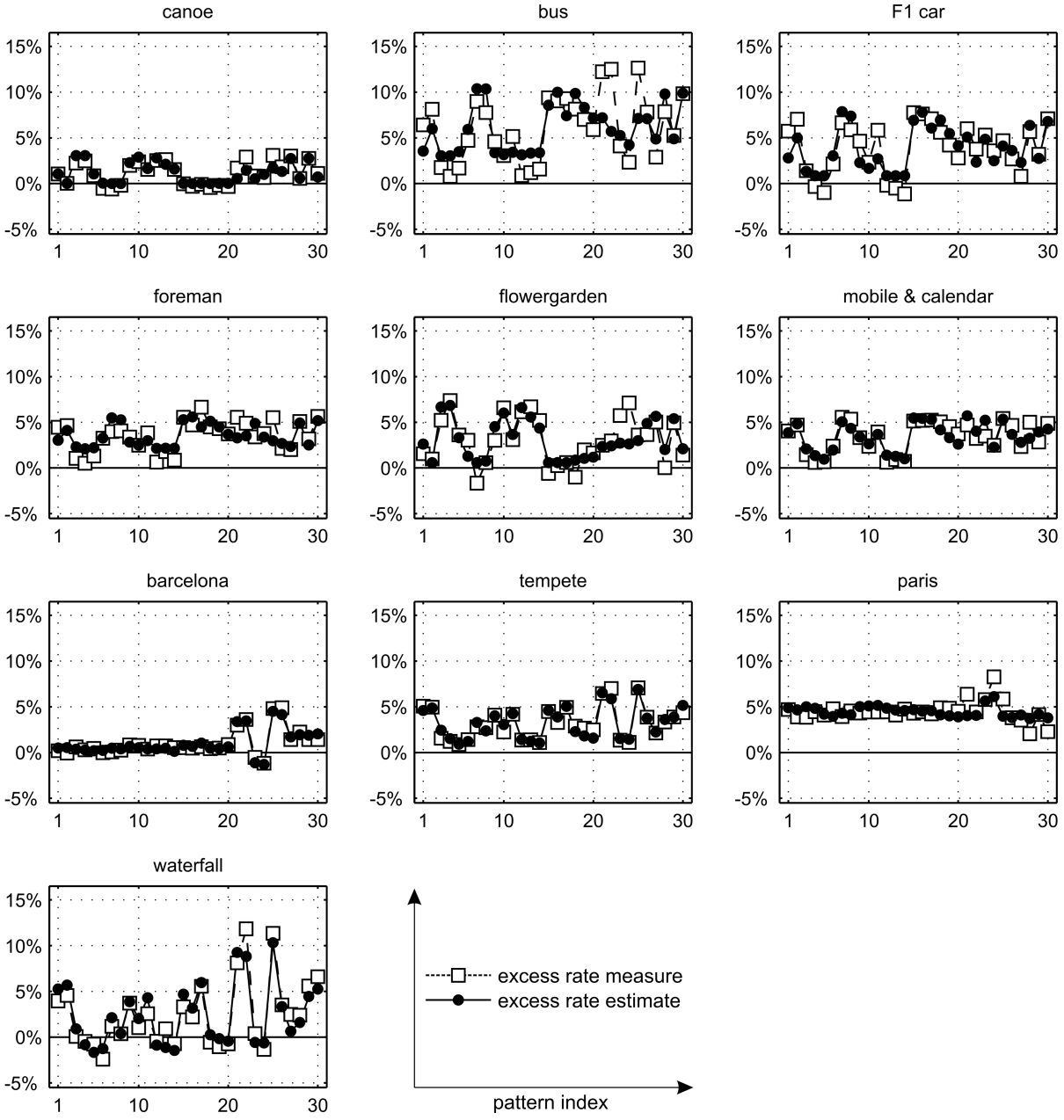


Figure 5.10: Simulation results for the proposed intra refresh algorithm sorted by descending motion activity. Estimation results are based on current motion are scaled to the reference measures range (minimum MMSE). (Results displayed for 35 *dB* simulations, other simulations see figures D.1 and D.2).

The effect of increased data rates caused by bad refresh patterns can be well observed with the *canoe* sequence, showing a camera pan to the right resulting in motion vectors pointing to the left. Both, the reference measure and estimates show that the patterns 2, 6, 7, 8 and 15 through 20 are well suited for this sequence. figure 5.6 confirms that these patterns proceed from the right to the left side. On the other hand, the worst-case pattern 3 proceeds against the global motion with maximum cost. Patterns 1 and 5 which also proceed against the global motion require less data rate, because at any instance of time only one border of a clean region must be protected instead of three borders. The

sequences *foreman*, *mobile & calendar*, *tempete* and *waterfall* prove that the cost estimation is also valid for complex motion scenarios.

Considering the simplicity of the underlying model, the estimates are remarkably accurate and show good correlation with the reference curves. However, the test sequences *bus*, *flowergarden* and *paris* show outliers in the estimate curves, which indicates that the chosen cost function does not consider all effects of the intra refresh mechanism. It is also possible that the average motion vector calculation is not precise enough in these cases because it favors large objects and loses information about small moving objects in the video sequence.

For the task of selecting an optimal refresh pattern, accurate estimates for low-excess rate refresh patterns are particularly interesting. Table 5.1 summarizes the simulation results for the optimum refresh patterns determined by the measurement and the estimate. In six out of ten cases the estimation algorithm finds the same optimum refresh pattern as indicated by the reference measures. In the case of the *mobile & calendar* and *F1 car* sequences, a small penalty of 0.1 % and 0.7 % excess data rate occurs because the estimation algorithm does not choose the optimum pattern. In the case of the *flowergarden* sequence the optimum pattern 7 is ranked as second by the estimation algorithm which leads to a significant excess data rate of 2.7 %. Similarly, the sub-optimal pattern selection for the *waterfall* sequence results in a 1.6 % excess data rate.

Table 5.1: Optimal refresh pattern selections.

sequence	Measures *)		Estimates *)	
	1st	2nd	1st	2nd
canoe	7 (-0.6 %)	6 (-0.5 %)	7 (-0.6 %)	8 (-0.2 %)
bus	4 (0.8 %)	12 (0.9 %)	4 (0.8 %)	3 (1.8 %)
F1 car	14 (-1.1 %)	5 (-1.0 %)	4 (-0.3 %)	5 (-1.0 %)
foreman	4 (0.5 %)	12 (0.6 %)	4 (0.5 %)	12 (0.7 %)
flowergarden	7 (-1.7 %)	18 (-1.0 %)	2 (1.0 %)	7 (-1.7 %)
mobile & cal.	4 (0.6 %)	12 (0.6 %)	5 (0.7 %)	14 (0.7 %)
barcelona	24 (-1.2 %)	23 (-0.5 %)	24 (-1.2 %)	23 (-0.5 %)
tempete	5 (0.8 %)	14 (1.1 %)	5 (0.8 %)	14 (1.1 %)
paris	28 (2.1 %)	30 (2.3 %)	28 (2.1 %)	30 (2.3 %)
waterfall	6 (-2.4 %)	24 (-1.3 %)	5 (-0.8 %)	14 (-0.8 %)
average excess data rate	-0.22 % (Δ=0.0 %)	0.12 % (Δ=0.34 %)	0.30 % (Δ=0.52 %)	0.24 % (Δ=0.46 %)

*) The first number is the pattern index, the second number in brackets is the excess data rate of the chosen refresh pattern.

Despite the fact that the optimum pattern was not chosen by the estimation algorithm in some cases, the simulations prove that always a relatively good pattern is chosen. The average excess data rates summarized in the bottom line of

Table 5.1 indicate that the refresh pattern selection based on the proposed estimation algorithm results in only $0.5\,\%$ excess data rate compared with the optimum pattern selection based on the measurements. Especially for the four test sequences with sub-optimal pattern selection, the possible penalty for selecting the worst refresh patterns lies in the range of $5\,\%$ to $10\,\%$ excess data rate. Therefore it can be concluded that the proposed estimation algorithm guarantees a good adaptation of the intra refresh to the global motion and avoids high excess data rates.

In appendix D the simulation results for the $30\ dB$ and $40\ dB$ operating points are shown in figures D.1 and D.2. The shape and matching of the estimates and reference curves are generally very similar. It is however remarkable that the excess data rate decreases towards higher quality. One reason for this effect can be found in the decreasing amount of skipped macroblocks when the data rate is increased. A bad refresh pattern therefore has more severe effects at low data rates because the skipped macroblocks are replaced by inter or intra-coded macroblocks with significant DCT data.

Summarizing the simulation results it can be concluded that the proposed algorithm is sufficiently accurate to choose a good refresh pattern with approximately $0.5\,\%$ average excess data rate at the $35\ dB$ operating point. Thus, introducing motion adaptation for the intra refresh efficiently minimizes the border protection cost and avoids bad refresh strategies which require up to $20\,\%$ more data rate than a data stream using standard IP coding for the $30\ dB$ operating point.

5.3 Real-time implementation of the adaptive intra refresh

This section describes the implementation of an intra refresh controller in a real-time video encoder using the estimation algorithm described in the previous section [86,87]. The description first discusses the frame-level flow chart and the functional blocks needed for the implementation. Afterwards the implementation flow chart of the estimation algorithm is presented.

5.3.1 Frame-level flow chart

The previous discussion of the basic estimation algorithm was evaluated in offline simulations using concurrent data and multiple encoding runs with different refresh strategies to verify the correctness of the proposed algorithm. In a real-time system it is necessary to modify the program flow as shown in figure 5.11.

The initialization of the algorithm is performed in steps 1) and 2). In step 1), the frame area is split into N_R regions and the mapping between the macroblocks and

regions is stored in a mapping memory. The initialization of the mapping memory which was used for the simulations in CIF resolution is given in table E.1. In practical implementations step 2) can be implemented with a table which is preloaded with the refresh patterns as static data. The initialization is given in table E.2.

The proposed implementation uses the motion data of the frames in the previous refresh cycle in order to estimate the best refresh pattern. Hence, the selection of the refresh pattern is based on old data and it is assumed that global motion is continuous during the refresh cycles. It is however advantageous to use only motion information of the last one or two frames of the preceding refresh cycle.

As the starting up of a real-time system is critical it is proposed to capture and process two consecutive frames at the beginning and perform the motion estimation for the step 3) of the algorithm on these two frames. The second frame can be encoded as a low-quality I-frame which is transmitted to allow instantaneous video display on the decoder. When the motion vectors are calculated for the second frame, the algorithm proceeds with step 4) and calculates the average motion of the regions. In step 5) the cost estimation of all refresh patterns is performed according to the estimation algorithm described in section 5.2.3.1 and the best refresh pattern is chosen in step 6). During the next N_R frames the refresh is performed with the selected refresh pattern. Hence, the cost estimation algorithm is only performed once every N_R frames. A detailed flow chart of the sequence level continuous intra refresh is shown in figure C.2 (appendix C).

In some applications it may be required to reduce the frequency of the intra refresh in order to decrease the additional data rate caused by the intracoded macroblocks. One possibility which is evaluated in section 5.9.3 is to change the number of regions, e.g. 6 or 24 (see also appendix E), to achieve higher or lower error robustness.

Another possibility to reduce the intra refresh frequency is to insert predicted frames without intra refresh regions when a refresh cycle is completed. This results however in a variable data rate stream which is usually not beneficial for low-latency applications. Therefore it is more advantageous to insert predicted frames (without intra refreshing a region) in between the frames of an ongoing refresh cycle. In this case, it is necessary to activate the border protection of clean regions in these P- frames which can lead to an increase in data rate.

Extending the refresh cycle over more frames generally increases the probability that global motion changes significantly and the selected refresh pattern becomes inefficient. Therefore, from the intra refresh strategy point of view it is desirable to have short refresh sequences as this lowers the border protection costs and reduces the probability of global motion changes.

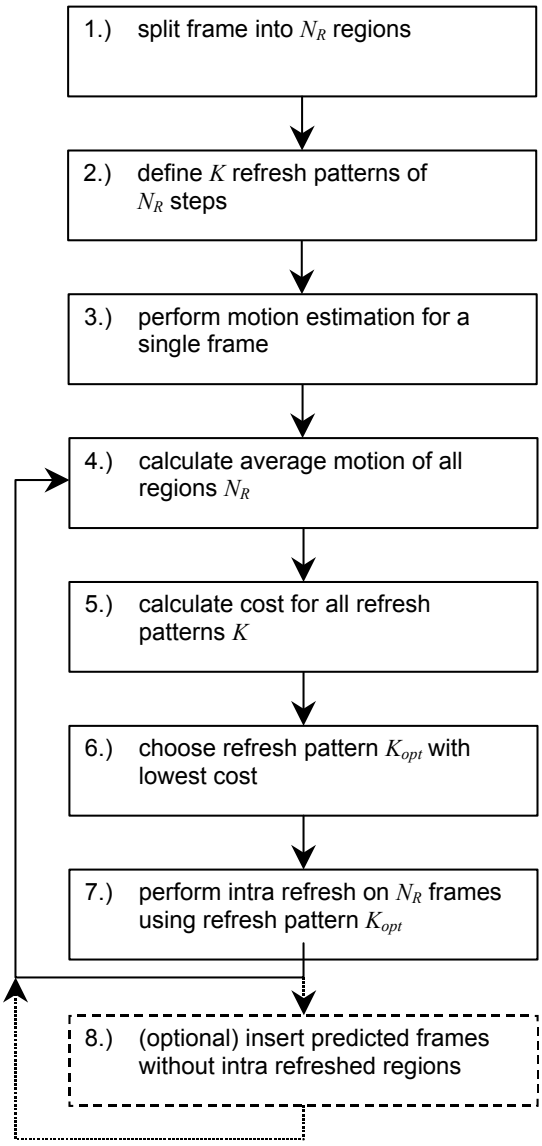


Figure 5.11: Frame-level program flow of a continuous intra refresh.

5.3.2 Structural overview

A structural overview of the intra refresh controller and its interface to a standard hybrid video encoder is shown in figure 5.12. The upper part of the figure shows the well known video encoder with the basic functional blocks for intra frame compression, temporal prediction and inter frame prediction. If the intra refresh controller is inactive, the mode decision unit can select between the output of the intra and inter compression units depending on certain measures. In standard operation mode, the mode decision will also generate the periodical I-frames by selecting the intra mode for a complete frame. The decompression unit generates the identical picture in the reconstructed frame memory as the decoder by using the intra/inter information and the prediction formed from the previous frame.

All functional blocks needed for implementing the adaptive intra refresh are realized by the intra refresh controller. During the encoding process of a predicted frame, all motion vectors and coding modes are stored in the motion vector memory. As every frame in an intra refreshed video sequence contains a section of intra-coded macroblocks without motion information, it is useful to store motion vectors of two or more preceding frames for estimating the average motion of regions. If energy consumption and computing power are of less concern it is also possible to calculate motion vectors also for intra-coded regions which are only used in the intra refresh controller.

When a refresh cycle is completed after N_R frames, the region motion estimator calculates the average motion of the regions based on the information stored in the motion vector memory. The individual macroblocks can be mapped into regions by reading the corresponding index in the mapping memory. A detailed description of a region motion estimator can be found in appendix C, figure C.1.

The result of the region motion estimation is a table which contains the information about the average horizontal and vertical motion of N_R regions. The distortion estimator uses this table to simulate the predefined refresh patterns which are stored in the refresh pattern memory according to the algorithm described in section 5.2.1. The output of this block is the index of the optimal pattern.

When the distortion estimate is finished, the encoding of the first frame of the refresh cycle can start. The intra/inter control reads the index of the current refresh region from the selected refresh pattern memory. During the encoding of a frame the intra/inter controller monitors the internal variables for macroblock position X and Y of the video encoder. A detailed flow chart of the controller is given in appendix, figure C.2. The macroblock coordinates are mapped through the mapping memory to regions and the region index of the current macroblock is compared with the index of the currently refreshed region in the refresh pattern memory. If the current macroblock is in a refresh region, the intra/inter controller forces an intra-coded macroblock by selecting the intra coding mode which overrides the internal mode control of the video encoder.

The intra/inter controller is also in charge of protecting clean regions against intrusion of dirty regions. This is achieved by signaling a restricted search range to the motion estimator when the currently encoded macroblock belongs to a clean region and is located on the border of an dirty region. Upon completion of a frame, the controller reads the next refresh region index and the process will continue until the refresh of the whole frame is completed.

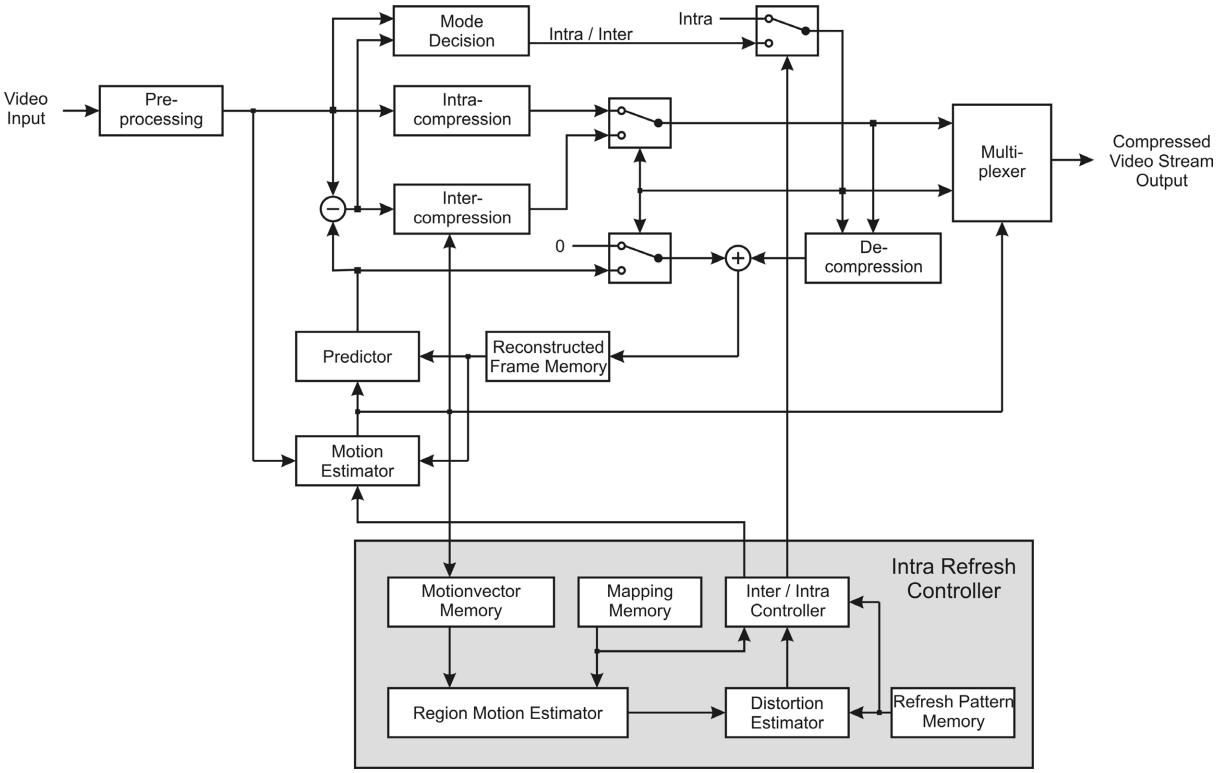


Figure 5.12: Structural overview of a hybrid video encoder with intra refresh controller using the proposed algorithm.

5.3.3 Estimation algorithm program flow

The estimation algorithm is described with two flowcharts. The main routine shown in figure 5.13 implements the outer two loops of the C_k cost calculation in equation 5.2 and determines the optimum pattern. The sub-function for the region cost measure which is shown in figure 5.14 performs the inner most loop of equation 5.2 and accumulates the $C_{k,n,i,j}$ for all neighbors of region i.

The main routine starts with initializing the variables *best_cost*, *best_pattern* and *current_pattern*. On the beginning of a pattern simulation in step 2), all regions are marked as "dirty regions" and the frame level simulation is initialized with zero cost in the variable *pattern_cost*. In the step 3), the frame level simulation is initialized with the *frame_cost* to zero and the region counter i to the first region. In step 4), the subroutine *region_cost* is called which returns the estimated distortion for the region i of a frame for the selected refresh pattern. The *frame_cost* is accumulated in step 5) and the region counter is incremented. The loop condition in step 6) will continue with step 4) until the cost of the complete frame was calculated. In step 7) the index i_{seq} of the region which was intra-coded in the current frame is determined by reading the refresh pattern memory and the corresponding region is marked as clean region. In step 8) the *frame_cost* is added to the *pattern_cost* and the simulated frame counter is incremented.

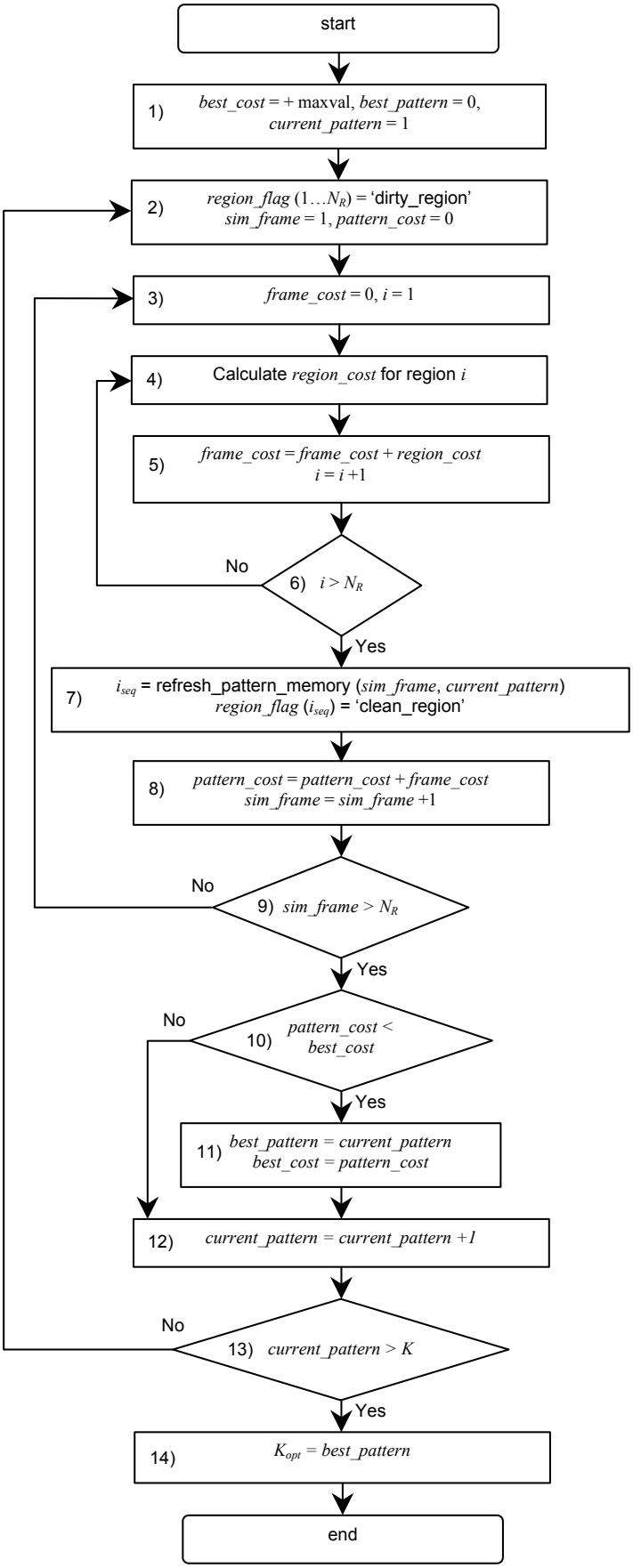


Figure 5.13: Best pattern selection algorithm.

When all frames of a refresh pattern are simulated, the loop condition 9) becomes true and the algorithm checks if the simulated pattern has a lower cost than the previously simulated patterns. If this is true, the algorithm continues with step 11) and stores the pattern index and the cost of the current pattern. In step 12) the *current_pattern* counter will be incremented and the loop condition for the pattern loop will be checked in step 13. When the estimation for all patterns was performed, the variable *best_pattern* returns the index of the pattern with the lowest estimated cost.

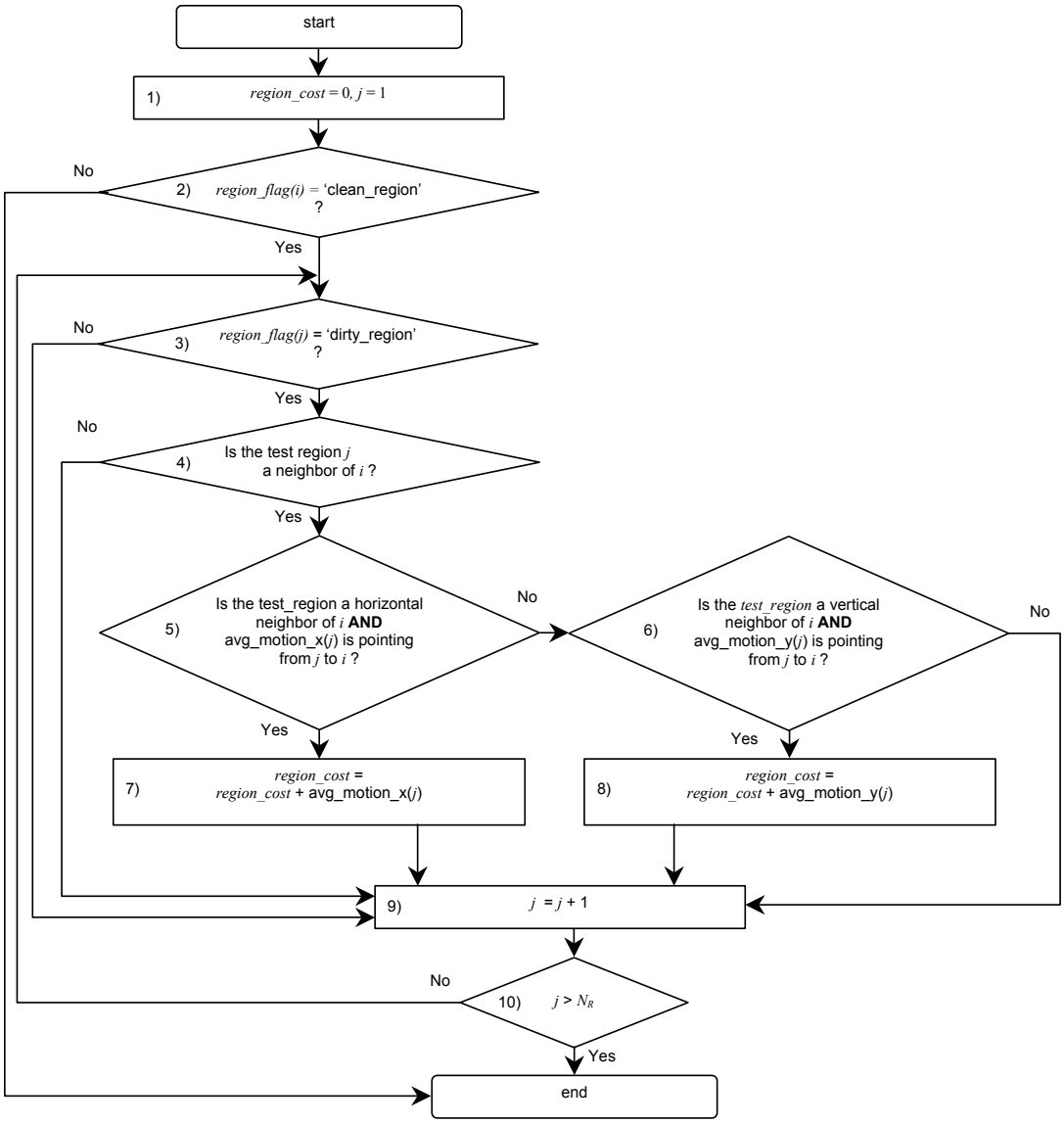


Figure 5.14: Region cost estimation.

The detailed flow chart of the cost estimation for region i is given in figure 5.14. Step 1) initializes the *region_cost* and the test region index j. For simplicity of implementation, the algorithm loops over all N_R regions of the frame and calculates their border protection cost with respect to the selected region i, if the two regions

are direct neighbors. If the currently simulated region is a dirty region, the test in step 2) fails and zero region cost will be returned because dirty regions are not protected and therefore do not contribute to the border protection cost.

If the selected region is a clean region, the algorithm continues with step 3) and verifies if the current test region is a dirty region. Only if this is the case, it must be verified if the regions are direct neighbors, otherwise the algorithm can proceed with the next test region. In step 5) it is tested if the two regions are horizontal neighbors and in this case the region cost is incremented by the average horizontal motion vector of the test region. Similarly it is tested in step 6) if both regions are vertical neighbors and in this case the region cost is incremented by the average vertical motion component of the test region. A detailed description of the average motion calculation is given in figure C.1 of the appendix.

If both tests in step 5) and 6) fail, it can be concluded that the test region j has no effect on cost of region i and the border protection cost is not incremented. In step 9), the test region counter j is incremented and the loop condition in step 10) will continue with step 3) until the maximum value is reached. If all regions were tested against the selected region i, the algorithm returns the value of $region_cost$ to the calling function.

5.4 Intra refresh error recovery and scene changes

Error recovery

The selection of the error recovery time has a significant impact on the data rate requirements as a higher number of intracoded frames or macroblocks increases the redundancy in the video stream. Considering an IP-mode video transmission, reducing the GOP-size successively approaches the coding efficiency and the instantaneous error recovery capability of the intracoded transmission.

Figure 5.15 illustrates the determination of the maximum error recovery time which is defined as the maximum time a display error is visible for the viewer. In the case of IP-coded video transmission all errors of the previous GOP are cancelled instantaneously when an error-free I-frame is received. Hence, the maximum error recovery time is equal to N_{GOP} frame periods.

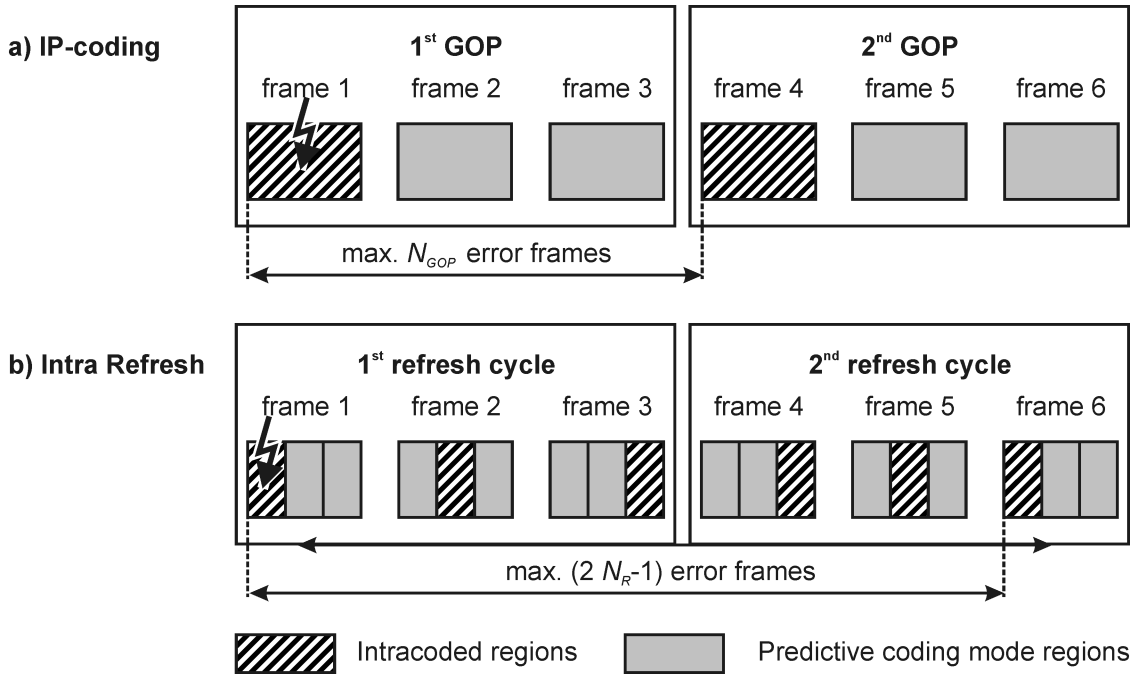


Figure 5.15: Worst-case error recovery for IP and IR coding modes.

For the case of IP and IPB video transmission the following rules apply for the average and worst-case error recovery time:

$$t_{recover,IP}^{max} = N_{GOP} \cdot T_{frame} \tag{5.3}$$

$$t_{recover,IP}^{mean} = (0.5\, N_{GOP} + 0.5) \cdot T_{frame}\,. \tag{5.4}$$

Considering the same GOP-size ($N_{GOP} = N_R$), the intra refresh coding modes generally have (N_{GOP} - 1) frames longer recovery times than the corresponding IP coding modes. This is caused by the sequential intra update during a whole GOP which replaces the instantaneous intra update of the IP coding mode. As can be seen in figure 5.15, a corrupted intra coded region can remain visible for almost two refresh cycles if the refresh pattern changes. Even if the same refresh pattern is applied continuously, an error in a refreshed region can spread into neighboring regions and the error cancellation is only guaranteed after one complete refresh cycle. Hence, the recovery times evaluate to

$$t_{recover,IR}^{max} = \left(2\, N_R - 1\right) \cdot T_{frame} \text{ and} \tag{5.5}$$

$$t_{recover,IR}^{mean} = \left(1.5\, N_R - 0.5\right) \cdot T_{frame}\,. \tag{5.6}$$

For achieving comparable error recovery performance, the intra refresh mode requires almost twice the intra macroblock rate as the standard IP coding mode. For the performance evaluation in section 5.9.3, IP, IPB and intra refresh coding modes are compared according to their worst-case error recovery time.

Scene-changes

The intra refresh coding mode is quite susceptible to scene changes which require a full-frame intra transmission. As the encoder must ensure that each frame generates approximately the same amount of data, scene changes as well as the initial start-up of the transmission may be handled effectively by one of the following methods:

- Large intra frame: In all common standards like MPEG-2, H.263 and H.264, large intra-coded frames which require more than one frame period for transmission are allowed in the low latency mode. In order to fulfill the real-time transmission requirements, several P-frames after the large I-frame must be skipped. Hence, the viewer will notice a short freeze of the video frame at a scene-cut and the temporal prediction of the first P-frame after the I-frame will be less efficient as the prediction spans over multiple skipped frames which reduces the correlation.

- Alternatively a low-quality intra frame can be transmitted at a scene-cut and the following frames successively add the missing texture information. Using an appropriate intra refresh strategy, the image will be reconstructed to full quality within the pre-defined interval of one refresh cycle.

- Similar to the low-quality intra frame, a sequential build-up of the new scene can be implemented while the last frame of the previous scene remains as background. In the development of the H.264 standard, "isolated regions" are proposed to support scene changes and intra refresh in the baseline profile [53,54]. Especially in the context of video conferencing applications, the proposed "box-out" method, which builds up the image sequentially from the center to the boundary, seems to be a good alternative.

It should be noted that in typical low-latency applications like video conferencing scene changes rarely occur, because there is only one stationary camera involved which outputs a continuous video stream. The temporal prediction may however be corrupted in some cases, e.g. if the camera is moved to another position or persons are moving through the scene close to the camera.

5.5 Low-latency rate control issues

In chapter 3 a detailed analysis of temporal and spatial data rate variations for different coding modes was presented and the effect on the delay in video transmission was analyzed in chapter 4. In order to reduce the buffer delays an intra refresh algorithm was developed which reduces the rate variations on frame-

level effectively. The basic algorithmic simulations in this document were performed without rate control means at fixed quantizer levels in order to avoid feedback on the intra refresh from a rate control algorithm. Hence, the intra refresh algorithm and the estimated latency bounds of section 4.5 are based on the "natural" data rate distribution of the video stream. This assumption is based on the fact that, in the case of high bitrate applications, the visual quality of an encoded video stream is maximized if the quantizer is constant throughout the whole video frame [77].

Nevertheless, a real-time system requires some means of adjusting the data rate generated by the encoder dynamically according to the complexity of the video content. Furthermore, a strict rate control is required to ensure the functionality of the system in cases where the "natural" data rate would violate the buffer constraints. This may be the case if more extreme video content than the selected video sequences is transmitted, or if lower latency requirements than the ones which were derived in section 5.9 must be fulfilled. While conventional rate control schemes usually employ a feedback control from the output stream buffer, the low-latency applications require tight buffer regulation which can be achieved by accurate rate modeling and possibly a look-ahead analysis of the current video frame.

Major publications in the field of rate control algorithms for low-latency applications date back to the years 1999 to 2001 and refer to the H.263 and MPEG-4 video standards. The conventional rate control methods of MPEG-2 [20] and comparable video standards determine the quantization parameter of the current macroblock by considering a simple feedback of the cumulative deviation of the actual bits needed for encoding the previous macroblocks from the target number bits. In order to satisfy the tight buffer constraints of low-latency applications, the following sequential method for calculating the quantization parameter can be applied as outlined in [78]:

1. modeling the relation between the number of DCT and syntax bits and the quantization parameter with a rate-distortion formula whose parameters are adaptively updated during the encoding process,

2. selecting the quantization parameter for the next macroblock according to the remaining bit budget of the current frame,

3. using the macroblock data statistics for rate-distortion modeling.

A basic rate control scheme for low-latency video encoders using the described method is presented in [77]. The rate control algorithm is based on modeling the relationships between macroblock complexity, quantization and the amount of data generated after the entropy coding stage. Using this model it is possible to

perform a pre-allocation of the available data rate of a frame which is then used for adapting the quantizer on a macroblock level.

The allocation of data rate is based on the variance σ of the prediction residuals of macroblock i

$$\sigma_i^2 = \frac{1}{A}\sum_{j=1}^{A+A'}(P_i(j) - \overline{P}_i)^2 \tag{5.7}$$

where $P_i(j)$ is the prediction error of the pixel j of A luminance and A' chrominance pixels. As the DC component of the prediction error is coded differentially between neighboring macroblocks, the average value is subtracted and only the energy in the AC-components is considered. Based on this energy measure, a quadratic model is used to estimate the number of bits B_i generated by a macroblock for a given quantizer value:

$$B_i = A\left(K\frac{\sigma_i^2}{Q_i^2} + C\right), \text{ with} \tag{5.8}$$

B_i = target number of bits for macroblock i,
A = macroblock size (e.g. 16^2),
Q_i = quantizer value for macroblock i,
K, C =model parameters.

The model parameters K and C are updated continuously during the encoding process. The variable C models the average number of bits needed for motion vectors and stream syntax, K is used to adjust the relationship between the macroblock variance and bits needed for coding the DCT coefficients.

The key equation for the model based rate control algorithm for the computation of the quantizer value is derived with Lagrange optimization. The quantizer value Q_i for the i-th macroblock of the current frame is calculated by:

$$Q_i = \sqrt{\frac{AK}{(B-ANC)}\frac{\sigma_i}{\alpha_i}\sum_{k=1}^{N}\alpha_k\sigma_k}\ , \qquad i = 1, \ldots, N \tag{5.9}$$

with N = number of macroblocks in the frame,
B = target number of bits for current frame,
α_i = distortion weight for macroblock i,

where the distortion weights α_i can be used to optimize the visual quality, e.g. by concentrating the available bit budget on certain regions of interest. The rate allocation strategy according to equation 5.9 will increase the quantizer of a macroblock with the square root of the macroblocks energy.

Based on the above equation a macroblock level rate control can be realized with the following basic calculations:

1. For the first frame of the sequence initialize:

   $i = 1,$ $N_1 =$ number of remaining macroblocks in the frame

   $C = C_1 = 0,$ $K = K_1 = 0.5.$

2. Initialize the remaining bit budget β_i to the target bits B for the current frame:

$$\beta_i = B \tag{5.10}$$

3. Determine and accumulate the variance of residuals for the complete frame

$$S_1 = \sum_{k=1}^{N} \alpha_k \sigma_k \tag{5.11}$$

4. Compute Q_i for macroblock i to meet the allocated target bit rate

$$Q_i = \sqrt{\frac{AK}{(\beta_i - AN_iC)} \frac{\sigma_i}{\alpha_i} S_i} \tag{5.12}$$

5. Determine the effective number of DCT bits $B^*_{Res,i}$ and the syntax bits $B^*_{Syn,i}$ used by macroblock i and update model parameters K and C with a weighted average based on measures of current macroblock:

$$K_i = \frac{B^*_{Res,i} Q_i^2}{A\sigma_i^2}, \qquad C_i = \frac{B^*_{Syn,i}}{A} \tag{5.13}$$

6. Update counters for remaining bit budget β_i and macroblock index:

$$\beta_{i+1} = \beta_i - B_i^* = \beta_i - B^*_{Res,i} - B^*_{Syn,i}, \quad N_{i+1} = N_i - 1 \tag{5.14}$$

7. Subtract the current macroblock variance from the accumulated energy value

$$S_{i+1} = S_i - \alpha_i \sigma_i \tag{5.15}$$

8. Continue with step 4. for next macroblock $i = i + 1$ until all N macroblocks of the frame are encoded.

In the case of very low delay requirements the encoding of macroblocks can be started when all video lines of the current macroblock rather than the complete frame are captured. Concerning the rate allocation it is however necessary to make some assumptions because the prediction residuals of the remaining macroblocks in the frame are not available. As the data rate of successive frames in a video sequence is in most cases quite stationary (figure 3.3) it can be assumed

that the energy distribution in the current frame is similar to the previous frame. Therefore, at the beginning of the frame, the S_1 value of the previous frame can be used and the remaining S_i values are continuously updated with data of the current frame. In the case of a dramatic scene change a linear rate distribution can be assumed for the first frame.

It should be noted that there exist several other methods to describe the rate-quantization relationship. The method as presented in [78] uses non-linear MMSE estimates instead of a closed equation and the macroblocks are classified according to their coding modes and statistical measures. In the case of low bitrate applications the ρ-domain which represents the number of non-zero DCT coefficients can be used for low complexity modeling of the rate-quantization as well as distortion-quantization relationship [79]. Finally a simple table-lookup method can be used [80].

5.6 H.264 Implementation aspects

The proposed intra refresh algorithm can be applied to various motion-compensated video codecs as long as the encoder can be forced to intra-code specific macroblocks and restrict the search range for other macroblocks. For implementing the intra refresh with the H.264 video coding standard some codec specific effects should be considered [30].

Single Slice Group Coding Method

The intra refresh method applied for the simulations was implemented standard-compliant in H.264 with all frames being encoded in one slice. Possible sources of error propagation into clean regions were identified in the intra prediction, the generic skip mode prediction, the sub-pixel interpolation and the deblocking filter.

The effects of intra prediction can be controlled with the constrained intra prediction flag which restricts the intra prediction of a macroblock to reference pixels taken only from other intra-coded macroblocks of the same frame. Therefore, the borders of intra-coded regions cannot be corrupted by stale data from dirty regions because predicted macroblocks are never taken as reference. The constrained intra prediction leads however to a reduced coding efficiency especially for isolated intra macroblocks.

One major source of possible error propagation was identified in skipped macroblocks of clean regions. In older standards like MPEG-2, a skipped macroblock used the original pixel values of the reference frame at the identical position. However, in the H.264 standard generic motion compensation based on a predicted motion vector (PMV) is applied for skipped macroblocks. The PMV uses

the motion information of three top and the left neighboring macroblocks, allowing higher compression of video sequences with continuous motion vector fields.

Due to the PMV mechanism, it is possible that errors propagate from dirty into clean regions if skipped border macroblocks have unfortunate motion vectors. Forcing the border macroblocks to non-skipped modes, i.e. intra or predicted, results in higher data rates for video sequences with many skipped macroblocks (e.g. low-bitrate streams or static scenes). For achieving optimal performance, it is necessary to check the PMVs of all border macroblocks and force a non-skipped mode only if the PMV would result in prediction from dirty into clean regions.

Two minor sources of error propagation into refreshed regions were found in the fractional sample interpolation of predicted border macroblocks and the in-loop deblocking filter. Both effects are generally minimized by the motion adaptive intra refresh because a good refresh sequence favors prediction from clean into dirty image regions.

As the H.264 standard uses a six-tap FIR filter with the coefficients [1, -5, 20, 20, -5, 1] for interpolating fractional half pixel positions, temporal prediction of clean border blocks can lead to small errors on the border pixels even if the primary source of prediction lies within the clean region. The strongest error occurs for the outer border pixel, if the motion vector is pointing to a half pixel position just inside the clean area. The worst-case error can be derived from the absolute values of the filter coefficients resulting in a maximum error of $(1+5)/(1+5+20+20) = 6/46 = 13\,\%$ for these pixels. The effect could be avoided by forcing the motion vectors of border macroblocks to integer pixel positions at the cost of a decreased coding efficiency.

Similar effects can be caused by the non-linear deblocking filter which adaptively smoothes edges of 4 by 4 pixel DCT blocks. For detecting potential block borders, the filter analyses the continuity of the neighboring four pixels on each side of a block and becomes active only if the step of the DCT block border is within a certain range which is most likely a blocking artifact and not an intended object border. If a block border is detected, up to three pixels on each side of the border can be modified. The strength of the deblocking process depends on a global strength parameter and the quantizer values of the involved macroblocks. As the deblocking filter cannot be deactivated on a macroblock basis, it is not possible to control the spreading of distorted pixel information from dirty regions into clean regions precisely when the deblocking filter is activated.

It is generally possible to reduce the visual effects of false subpixel filtering and deblocking filter by introducing additional intelligence in the decoder. By tracking the intra update of the video sequence, the decoder can detect cases where the prediction or deblocking routines access stale data which is not available (i.e. at

startup) or corrupted by transmission errors. In these cases, the subpixel interpolation could duplicate the border pixels of the clean region instead of using dirty regions pixels and the deblocking filter could simply be turned off along critical outer borders of clean regions. Both counteractions would lead to small drift effects between the encoder and decoder but this approach would help to conceal the errors effectively.

Multiple Slice Groups Coding Method

In the H.264 standard additional tools for error protection and concealment were introduced which are summarized in [30]. In the context of the desired intra refresh, the concept of slice groups and flexible macroblock ordering are of special interest. Using these features, the problems concerning side effects of subpixel interpolation and the deblocking filter can be selectively avoided on the borders of clean regions.

The slice groups introduced in H.264 allow a flexible grouping of macroblocks into up to eight different groups which can be encoded and decoded independently to the largest degree. The grouping of macroblocks into slice groups is signaled in the picture header, allowing to choose from either several generic mappings or a completely user-defined mapping.

In order to achieve a gradual intra refresh, [53] introduces the concept of isolated regions consisting of at least two independent slice groups. The foreground slice groups correspond to the clean regions that were already refreshed, whereas the leftover slice group corresponds to the dirty regions. The foreground slice groups can evolve and grow over a sequence of frames in several pre-defined manners as described in more detail in [81,82].

Comparing the isolated regions approach with the region-based intra refresh proposed in this document, the usage of square slice groups seems to be the most natural and efficient way for implementing the described refresh strategies. Although it would be possible to generate the proposed intra refresh patterns by combining up to eight rectangles of different sizes, it is not desirable to have more than two slices or slice groups as this would reduce the coding efficiency due to the constrained inter prediction.

On the other hand it is possible to realize a completely user-defined macroblock to slice group mapping which offers the flexibility needed for the evolving region-based intra refresh. The signalling is defined in the H.264 standard as “explicit slice group map type” (*slice_group_map_type* = 6) and described in more detail in [82]. In the case of two slice groups, the signalling requires one bit per macroblock, resulting in roughly $40\,kbit$ data rate for a SDTV video sequence. Hence, the overhead for the signalling is roughly $4\,\%$ for a $1\,Mbit$ data stream.

In order to reduce the signalling overhead, it would be desirable to modify the refresh patterns in a way that the signalling in H.264 can be implemented more efficiently. Thus, the clean regions should grow to squares in every frame rather than having arbitary arrangements of multiple rectangles.

The usage of slice groups eases the controlling of border effects between dirty and clean regions. The parameter *disable_deblocking_filter_idc* can be used for disabling the deblocking filter on the edge of the clean regions. Concerning the inter prediction process no further precautions other than the restriction of the motion search range of clean regions must be taken. The subpixel interpolation is handled on slice borders similarly as frame borders without crosstalk from other slices. Some further parameters in the *motion constrained slice group set* can be used to support the decoder in interpreting the intra refresh correctly.

5.7 Intra refresh coding mode latency analysis

5.7.1 Frame-CBR intra refresh

The frame CBR intra refresh is a sub-optimal implementation of the proposed intra refresh principle which minimizes the delay at the cost of at least 15 % increased data rate compared with the IP coding mode. The reduced efficiency is mainly caused by the data rate variations of the embedded intra-coded regions which depend on the local texture. In contrast to the previous assumption that all frames of an IR encoded video sequence have approximately the same size, for some test sequences strong variations of the frame data rate can be seen.

The effect can be well observed on the *flowergarden* test sequence in the first column of figure 5.18 (chapter 5.7.2) where the data rate increases when the lower section of the frame is refreshed. In each frame an intra-coded section of approximately the same size is transmitted while the remaining macroblocks of the frame are encoded in predictive mode. When the refresh propagates in groups of macroblocks from the top-left to the bottom-right section as illustrated in figure 5.5, the amount of intra-coded texture increases towards the end of the refresh cycle.

In order to achieve optimal performance, the rate variations should be averaged out over a complete refresh cycle (GOP-CBR operation) as discussed in the next section. Nevertheless, it is also possible to achieve a constant bit rate per frame (frame-CBR) if additional transmission bandwidth is allocated for the largest frames as calculated in table 5.2. Because the “small” frames with little intra-coded texture information require less data rate than “large” frames, the remaining transmission capacity cannot be used. It should be noted that an optimized buffer strategy which allows spreading excess data of large frames over

previous and successive frames will asymptotically result in the GOP-CBR intra refresh described in the next section.

The overhead data rate for frame-CBR is calculated by the ratio of the highest frame data rate of the sequence with respect to the average frame data rate. The simulations indicate that about half of the test sequences require less than 10 % overhead data rate for frame-CBR transmission compared with optimal IR or IP coding modes. However, the other sequences require high overhead data rates up to 38 % which results in an average overhead data rate of 14.6 %.

In any case, the intra refresh transmission with frame-CBR allows about 50 to 85 % rate savings ($N_{GOP} = 12$) compared with the intra mode transmission.

If a faster recovery time is required, shorter refresh cycles can be used with a higher amount of intra-coded data. In the bottom section of table 5.2, the resulting data rates for short 6 and 3 frame intra refresh cycles as well as a long 24 frame refresh cycle are listed. Details of the applied refresh groups and refresh patterns are documented in appendix E. The average of the resulting overhead data rate is typically in the range of 9 % to 14 %.

Table 5.2: Frame-CBR intra refresh overhead data rates.

GOP size	Test sequence	Average frame rate data [bpp]	Max. frame data rate [bpp]	Min. frame data rate [bpp]	IR frame CBR overhead
12 frames	canoe	0.489	0.507	0.467	3.6 %
	bus	0.343	0.374	0.313	8.9 %
	F1 car	0.300	0.319	0.266	6.0 %
	foreman	0.140	0.193	0.118	37.6 %
	flowergarden	0.505	0.642	0.417	27.1 %
	mobile & calendar	0.632	0.684	0.590	8.2 %
	barcelona	0.614	0.649	0.575	5.8 %
	tempete	0.444	0.474	0.412	6.8 %
	paris	0.248	0.305	0.208	23.0 %
	waterfall	0.126	0.150	0.102	18.7 %
	average	0.384	0.430	0.347	14.5 %
3 frames	average	0.637	0.688	0.586	8.8 %
6 frames	average	0.469	0.526	0.409	14.0 %
24 frames	average	0.342	0.372	0.3194	12.0 %

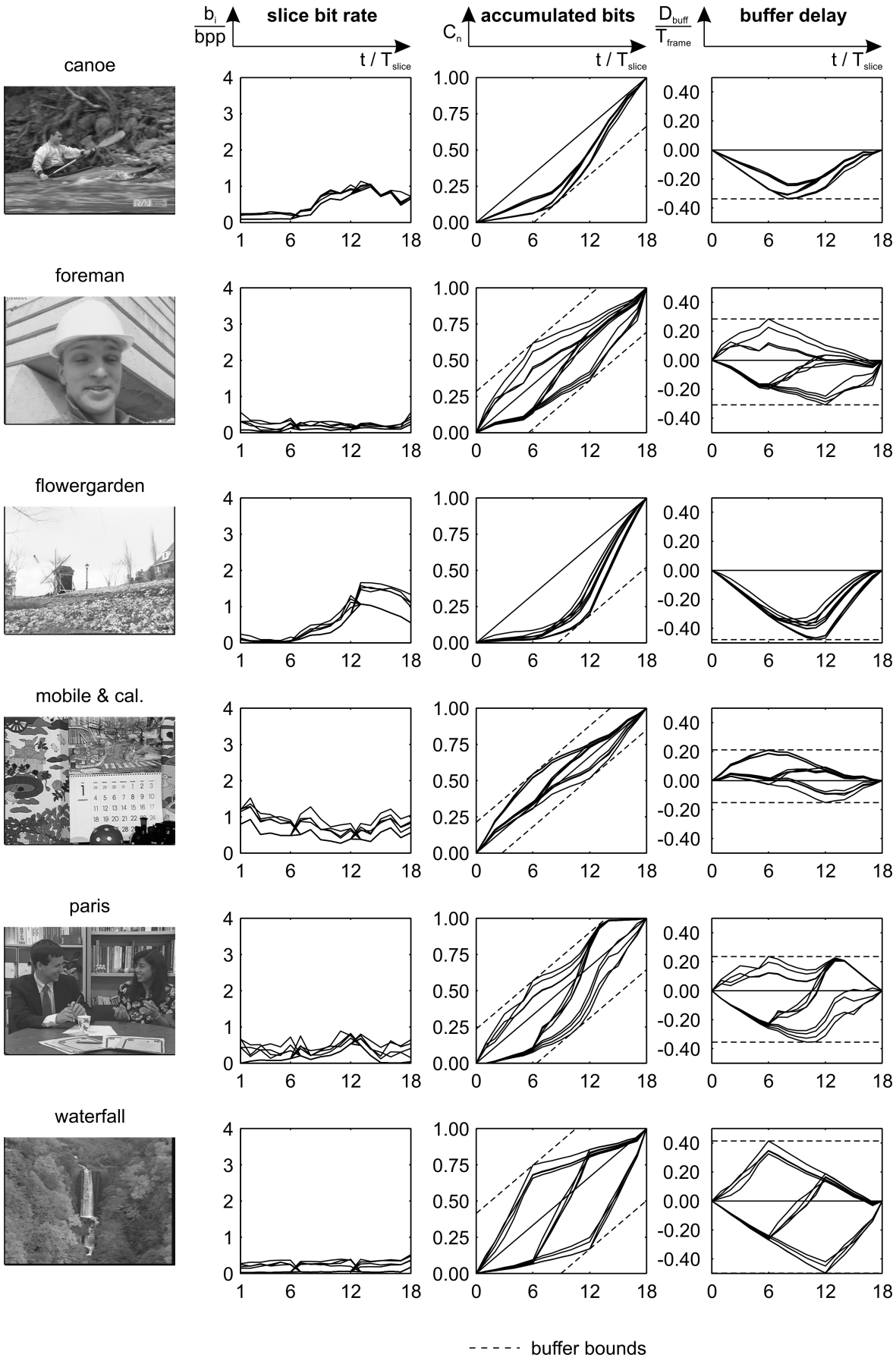


Figure 5.16: Frame-CBR intra refresh buffer analysis (slice resolution). The 12 individual curves per plot show the buffer characteristics for the 12 frames of a refresh cycle.

Due to the fact that the overhead data rates of different video sequences have a large variance, the significance of the calculated mean values should not be overrated. The roughly 9 % to 15 % overhead data rate are more likely a lower bound and it is quite likely that 20 % to 30 % overhead data rate are required for this coding mode. Implementing this coding mode certainly would require in-depth analysis and optimization for sequences with very high frame size variations.

The buffer analysis shown in figure 5.16 is based on the assumption that all compressed frames have the same size. Hence, the unused transmission bandwidth of small frames is linearly stuffed with filling data according to the local data rate. This is probably not an optimal strategy, but it is good enough to derive upper bounds for the buffer delays of this coding mode.

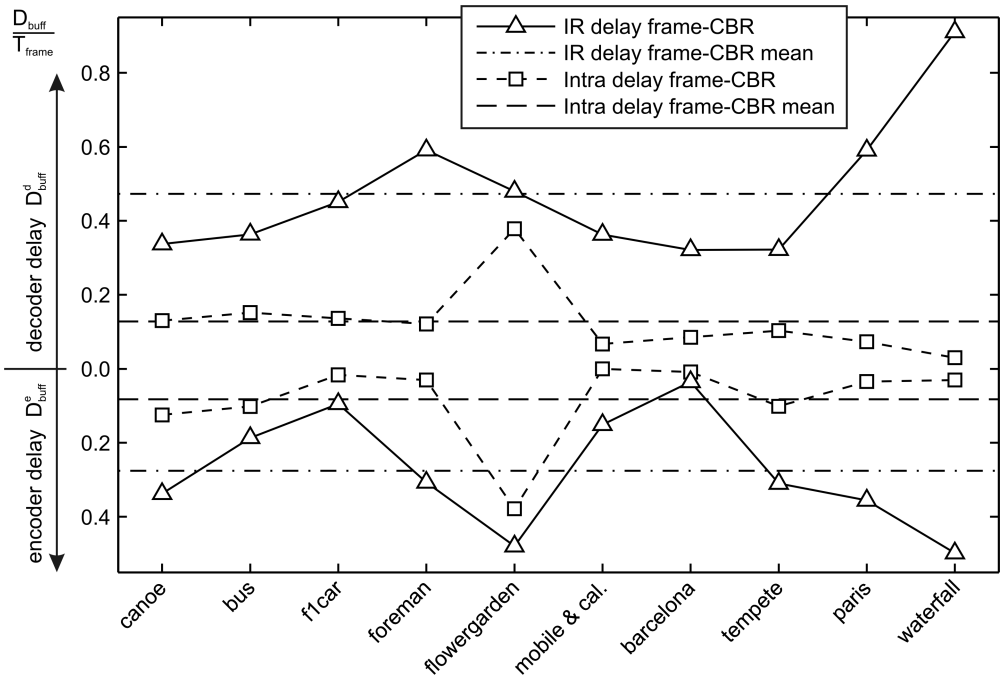


Figure 5.17: IR frame CBR delay analysis (N_{GOP} = 12 frames, slice resolution simulation, details table B.7).

The bit rate deviation analysis in the third column of figure 5.16 has different characteristics depending on the predictive coding gain and the local rate variations. On the one hand, the sequences *foreman*, *paris* and *waterfall* with high predictive coding gain show a symmetrical buffer delay. This behavior can be explained by the concentration of data rate in the intracoded frame sections which gather nearly 50 % of the data rate of a complete frame. On the other hand, the *flowergarden* and *canoe* sequence have a strong bias towards encoder buffering similar to the intra coding mode shown in figure 4.17. The reason for this can be found in the lower predictive coding gain which reduces the buffer effects of the intra-coded regions. Furthermore, the rate distribution of intra-coded and

predicted frame segments are quite similar and biased towards the bottom of the frame.

The delay analysis in figure 5.17 shows that the frame-CBR intra refresh can be implemented with a total buffer delay of $\text{mean}(D_{buff}) = 0.75$ to $\max(D_{buff}) = 1.4\ T_{frame}$ which is significantly above the intra coding mode delay. Nevertheless, the slice resolution analysis indicates that the buffer delay can be set significantly below the conservative $2\ T_{frame}$ worst-case assumption for frame-CBR mode.

The delay requirements for the individual sequences are quite different and show only little correlation with the intra coding mode delay. It is also remarkable that the *waterfall* sequence which has very balanced texture resulting in low buffer delays for the intra coding mode, requires the highest buffer delays in frame-CBR intra refresh mode. On the other hand, the high buffer requirements of the *flowergarden* sequence for the intra coding mode also result in high buffer requirements in IR coding mode. It can be concluded that the delay of frame-CBR intra refresh is susceptible to high predictive coding gains as well as unbalanced texture.

It should be noted that the sequence in which the regions are refreshed does not affect the latency characteristics of the frame-CBR IR mode. Therefore, the optimization algorithm proposed in section 5.2 can be applied without any restrictions and there is no need to consider possible effects of border protection in the buffer latency analysis. It was also verified that the slightly different size of the refresh regions (see figure 5.5) has only a minor effect on the delay calculation.

5.7.2 GOP-CBR intra refresh

In this section the buffer and delay requirements of the region based intra refresh method with GOP-CBR are examined. In a first step the frame resolution delay simulations are presented, which helps to explain the effects of the subsequent slice resolution delay analysis. The delay analysis results form the basis of the joint data rate and delay optimization which will be presented in section 5.8.

The significant overhead data rate of the intra refresh with frame-CBR can be eliminated if the CBR interval is extended to a complete intra refresh cycle. For the simulation of the GOP-CBR encoding schedule, all 30 variants of the predefined intra refresh patterns (figure 5.6) are simulated and collected in a single plot. The delay and buffer bounds are derived from the worst-case buffer requirements of all 30 refresh patterns, assuming that a constant bit rate per GOP can be achieved. It should be noted that the buffer analysis presented in this section does not consider the effects of border protection as discussed in section 5.2 because this would have required excessive manual data import from 3600 individual frames.

The detailed buffer simulation results are shown in figure 5.18. Analyzing the accumulated bits plot it can be seen that the buffer requirements for the selected test sequences are always smaller than for the IP coding mode shown in figure 4.17. Furthermore, the IR encoding schedule is more symmetrical compared with the strong bias on decoder buffering in the IP coding mode. In the third column the buffer delay of the two patterns which define the maximum encoder and decoder buffer delay are colored in green and red.

A summary of the IR buffer delay requirements for the frame resolution simulation is shown in figure 5.20. The mean value of the delay is $2.92\ T_{frame}$ ($D^{e}_{buff} + D^{d}_{buff} = 1.29\ T_{frame} + 1.63\ T_{frame}$), maximum delay is $4.27\ T_{frame}$ ($1.74\ T_{frame} + 2.53\ T_{frame}$). The worst-case delay is defined by the *flowergarden* sequence as in the intra coding mode. This sequence also has the highest variations of frame sizes due to the unbalanced amount of texture in the frames. Compared with the IP frame resolution simulation results, the worst case delay is reduced by 52 percent from $8.82\ T_{frame}$ to $4.27\ T_{frame}$ and the mean and median values by 45 % and 39 % respectively.

The delay optimization for the IR coding mode can be further improved if a slice resolution analysis is applied which considers the sub frame-level data rate variations. The relevant simulations are displayed in figure 5.19 and figure 5.20. The patterns which define the maximum buffers are highlighted in red and green.

Considering column 1 in figure 5.19, it can be noticed that the slice resolution data rate for refresh pattern 1 has an oscillating amplitude. The high-data rate sections are generally slices which include intra-coded regions, while two thirds of the slices which do not include intra-coded regions have lower data rates. The oscillations caused by intra refresh regions also appear in the accumulated bits plots but they are overlaid by a second effect. The positive and negative bending of the curves is caused by the different frame sizes due to different amounts of intra texture as discussed in the frame-level analysis.

Taking a closer look at the bit rate deviation plots in the third column reveals that different intra refresh patterns result in different bending of the curves. Hence, the delay requirements depend on the selected intra refresh pattern and an optimization of the delay should be possible by selecting the refresh pattern with the lowest buffer requirements.

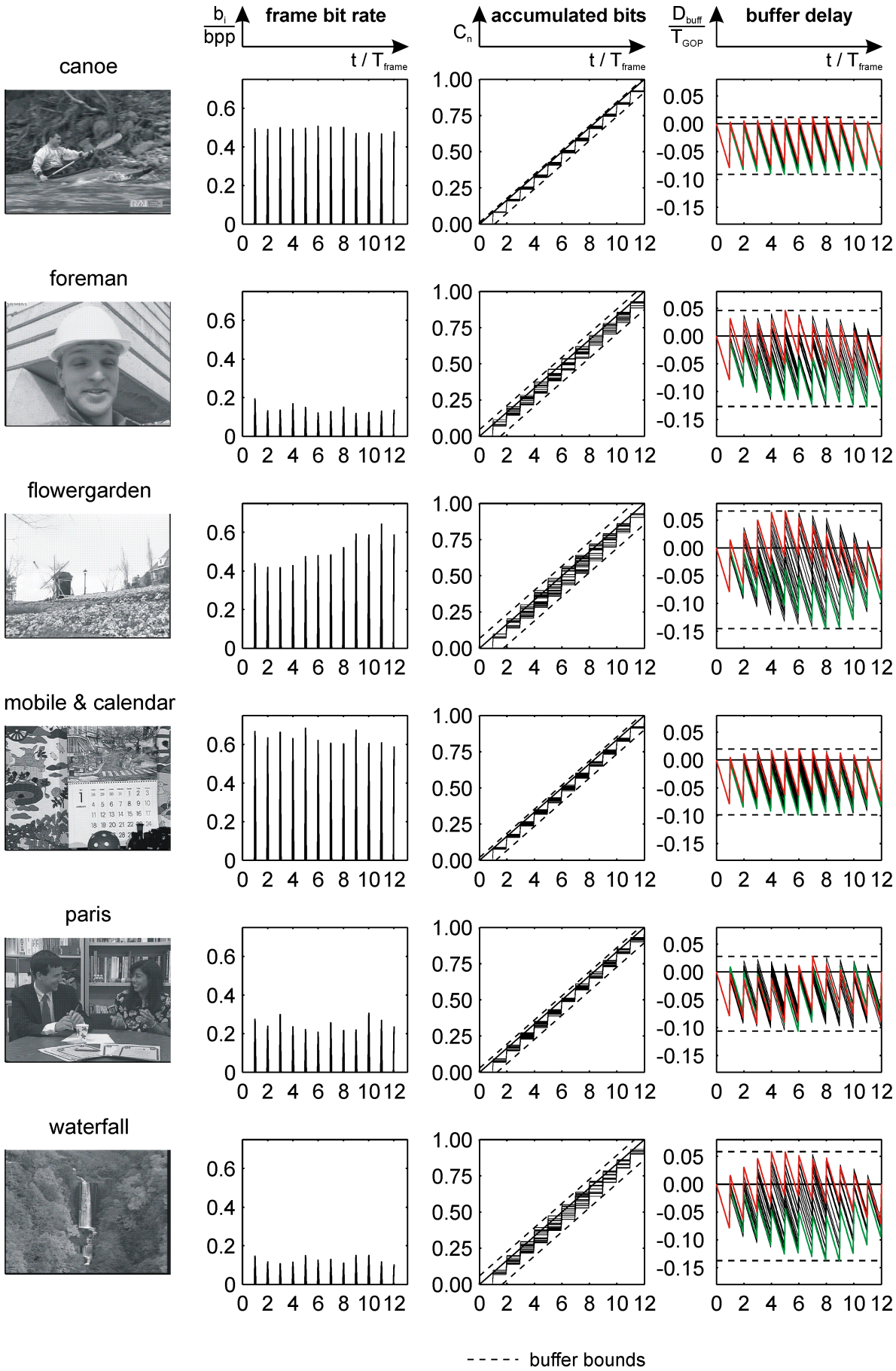


Figure 5.18: GOP-CBR intra refresh buffer analysis (frame resolution). Column 1 shows only pattern 1, column 2 and 3 show all 30 refresh patterns.

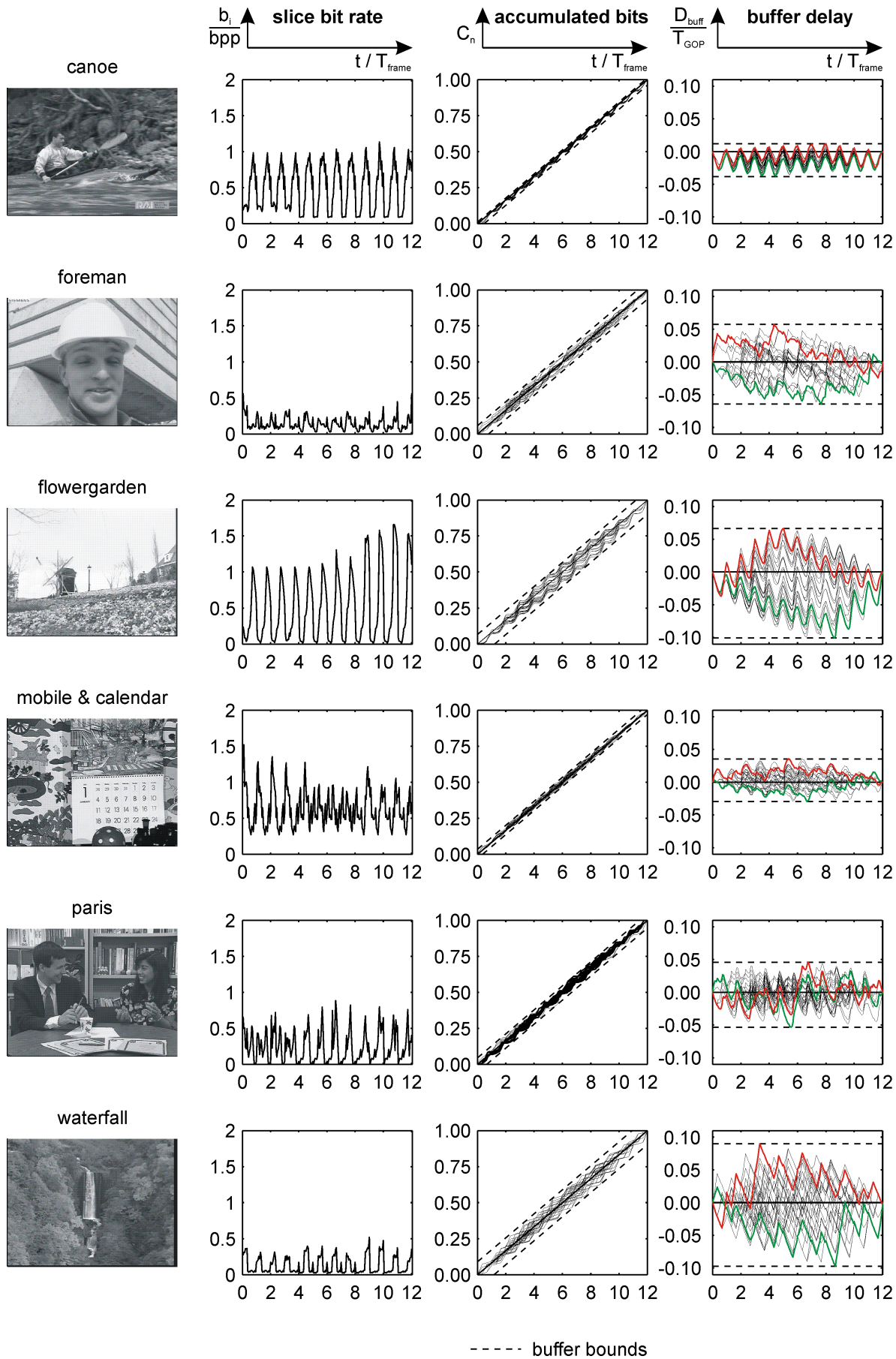


Figure 5.19: GOP-CBR intra refresh buffer analysis (slice resolution). Column 1 shows only pattern 1, column 2 and 3 show all 30 refresh patterns.

The summary plot shown in figure 5.20 therefore includes the worst-case delay bounds derived for the 30 intra refresh patterns and the best-case delay bounds for the refresh pattern with lowest delay (l-opt). For the standard mode, the worst-case delay bounds are defined by the two patterns with maximum encoder and maximum decoder buffer requirements whereas the latency-optimized mode chooses the pattern which minimizes the sum of encoder and decoder buffer requirements. Considering the worst-case delays it can be noticed that the *flowergarden* and *waterfall* test sequences require the largest buffers resulting in a maximum buffer delay of 3.45 T_{frame} (1.21 T_{frame}+ 2.24 T_{frame}). The mean value delay of the buffer delay is 1.67 T_{frame} (0.58 + 1.09 T_{frame}). Compared with the IP slice resolution simulation results, the maximum delay is reduced by 52 percent from 7.21 T_{frame} to 3.45 T_{frame} and the mean and median values by 46 % and 48 % respectively. Hence, the relative delay reduction from slice and frame mode simulations are almost identical.

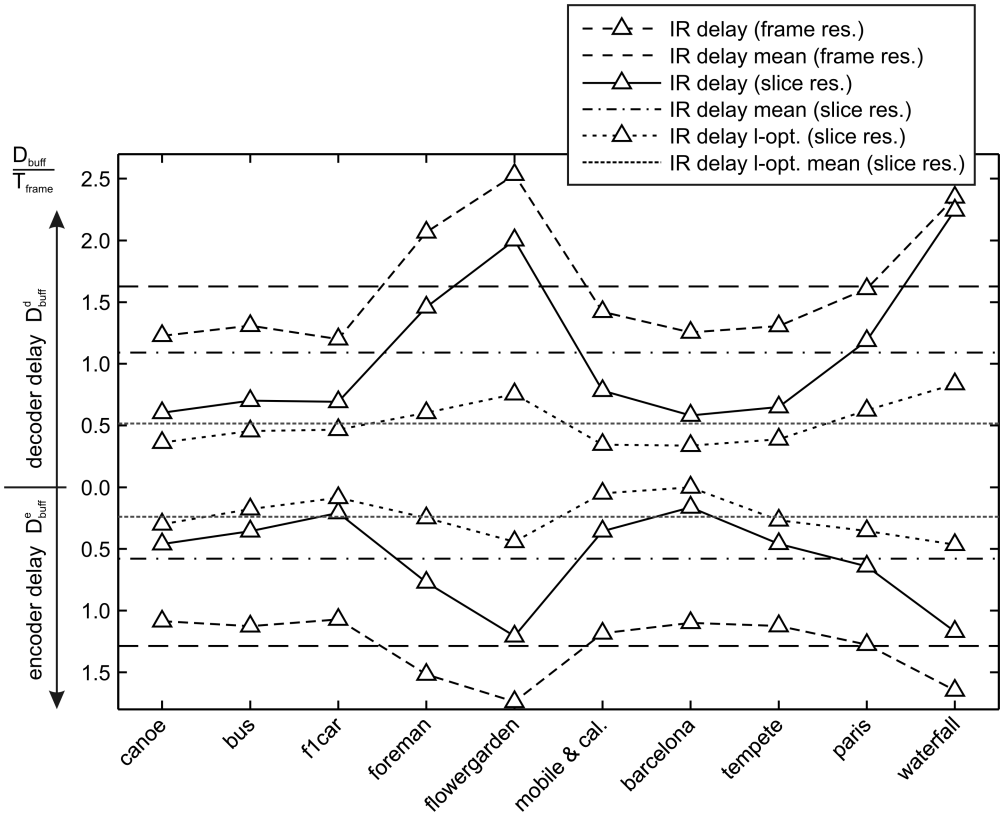


Figure 5.20: Intra refresh delay analysis with standard and latency-optimized (l-opt.) pattern selection (details in tables B.8, B.9, and B.10).

Comparing the intra refresh delay results with the intra mode (figure 4.16), it can be seen that the *flowergarden* sequence is a worst-case candidate in both coding modes. Furthermore for the IP-coding mode (figure 4.18) a similarity of the worst-case sequences *waterfall* and *foreman* can be found. It is however remarkable that in both cases the delay overhead compared with the other sequences is reduced significantly. In the case of the *flowergarden* sequence, the intra mode delay is 0.76/0.21 = 3.62 times above the mean value (median: 0.76/0.15 = 5.07) which is

reduced to an overhead of $3.21/1.67 = 1.92$ (median: $3.21/1.20 = 2.67$) in the intra refresh mode. Also for the *waterfall* sequence, the delay overhead is reduced from $6.97/3.10 = 2.25$ (median: $6.97/2.33 = 2.99$) in IP-mode to $3.41/1.67 = 2.04$ (median: $3.41/1.20 = 2.85$) for IR coding mode (numbers taken from appendix B).

A detailed analysis of the intra refresh delay simulations reveals that the 30 predefined refresh patterns can result in significantly different buffer requirements for the same test sequence. This effect can be observed in the buffer delay plots of figure 5.19 (third column) where the minimum and maximum buffer delays are usually defined by only one extreme refresh pattern. Especially for the high-latency sequences *waterfall*, *flowergarden* and *foreman* it can be seen that refresh patterns with relatively small positive and negative extremes exist.

The resulting delays for latency-optimized (l-opt.) refresh patterns are shown in figure 5.20 and table B.10. Selecting a latency-optimized intra refresh pattern is of special interest for the high-latency sequences. Considering the encoder buffering, the maximum delay of the optimal refresh pattern can be reduced to $0.47\ T_{frame}$ which is even 20 % below the mean of the non-optimized encoder buffer delays. Especially for the test sequences *flowergarden* and *waterfall* which have the highest delays in non-optimized IR mode, the latency-optimized refresh patterns achieve delay reductions of more than 60 %. Similar results can be achieved for the decoder buffering, where a delay of $0.84\ T_{frame}$ is sufficient for all test sequences. Hence, for the optimum intra refresh patterns a buffer delay of $1.31\ T_{frame}$ ($0.47\ T_{frame} + 0.84\ T_{frame}$) is sufficient in all cases. Compared with the non-optimized intra refresh the maximum buffer delay is reduced by 62 % (from $3.45\ T_{frame}$ to $1.31\ T_{frame}$) and the mean value by 54 % (from $1.67\ T_{frame}$ to $0.76\ T_{frame}$).

5.8 Joint data rate and latency optimization

In this section the results of the intra refresh selection algorithm and the intra refresh delay analysis of the previous section are combined to achieve optimal system performance. It is the goal of this analysis to show the feasibility of joint intra refresh optimization which achieves minimum delays with highest video quality. The joint optimization makes use of the quasi-redundancy of the 30 predefined intra refresh patterns. Based on the fact that for every video sequence more than one refresh pattern can be found which results in a negligible excess data rate, a second optimization pass can select the delay-optimized pattern out of these candidates.

The delay analysis in section 5.7.2 shows that it is possible to select optimum refresh strategies for critical video sequences which allow system delays below

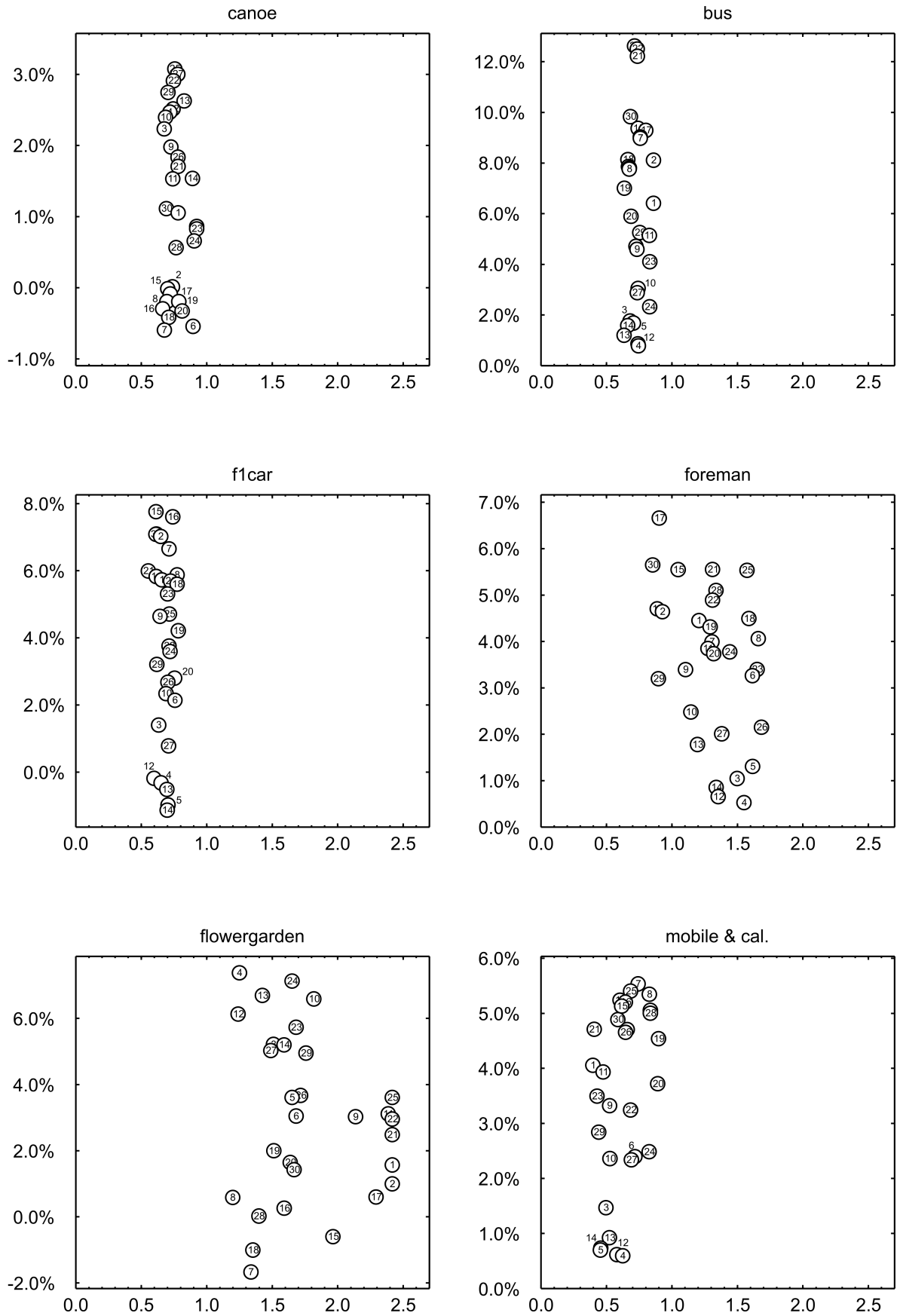
1.5 T_{frame}. This analysis is based on a slice resolution simulation of the temporal data rate variations of the encoded video sequence.

In figure 5.21 the results of the delay analysis and the excess data rate measurements are combined for the joint optimization. The delay requirement of each individual sequence is derived by selecting the maximum value of the encoder and decoder buffer delays. It should be noted that the x-axis scaling of the delay is equal for all sequences, while the y-axis scaling of the excess data rate is adjusted per sequence for better visibility.

As a first observation it can be noticed that the delay spread for different refresh patterns on the same test sequences is usually below a factor of two, while the delays between different sequences vary by a factor of 5 between 0.5 T_{frame} and 2.5 T_{frame}. The detailed analysis of the plots shows that most of the test sequences cannot meet a buffer requirement below 0.5 T_{frame} at all. On the other hand, it can be seen that six test sequences can meet a 1.0 T_{frame} buffer requirement for all refresh patterns while the remaining four test sequences can meet a 1.0 T_{frame} to 1.3 T_{frame} buffer delay only for some refresh patterns. It is therefore of special interest to examine the test sequences *foreman*, *flowergarden*, *paris* and *waterfall* in detail.

With the *foreman* sequence only the refresh patterns 12 and 14 can meet a 1.4 T_{frame} delay bound while maintaining a low excess data rate below 1 %. A lower delay could be achieved with other refresh patterns, where the patterns 13 and 10 are the most promising candidates with an excess data rate of 1.78 % and 2.48 % respectively. The *flowergarden* test sequence shows the largest delay spread from 1.2 T_{frame} to 2.5 T_{frame}. Out of the 30 candidates, the four refresh patterns 7, 18, 28 and 8 satisfy a 1.4 T_{frame} delay criterion with excess data rates below 0.6 %.

In the case of the *paris* test sequence, more than half of the refresh patterns achieve 1.4 T_{frame} or better delays. However, the majority of sequences requires excess data rates of 4 % to 5 %, but the optimum refresh pattern 28 achieves the lowest excess data rate and very low delay. Similarly, for the *waterfall* sequence the refresh patterns 23 or 10 can be selected which satisfy a 1.4 T_{frame} buffer constraint with low excess data rate.



(Figure continued on next page.)

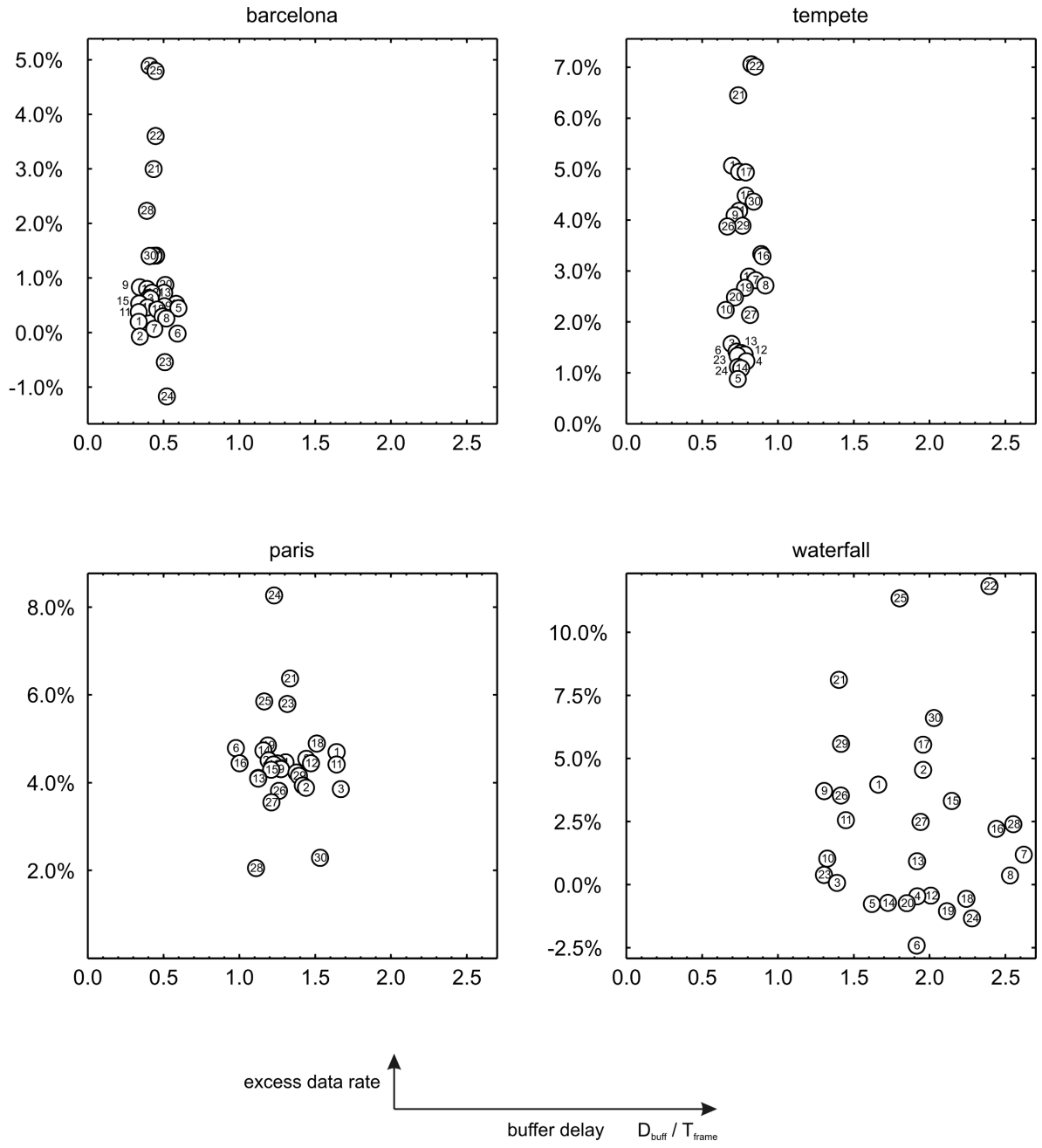


Figure 5.21: Refresh pattern excess data rate vs. buffer delay estimation.

The discussion clearly shows that joint optimization of quality and delay can be achieved for the region based intra refresh. Based on these observations a joint optimization algorithm can be developed which also considers long-term texture and motion variations of the video sequence. This research is however out of the scope of this document as it involves a more detailed analysis and optimization of the selection algorithm considering performance issues of varying video content.

5.9 Coding mode performance summary

5.9.1 Buffer latency comparison

In this section the buffer analysis results for the different coding modes are summarized for slice resolution simulations. The three characteristic values maximum delay, mean delay and median delay are shown in figure 5.22 for each coding mode for a GOP-size of $N_{GOP} = 12$ frames. The total delays are calculated by adding the corresponding average values of the encoder and decoder buffer delays. It should be noted that all IP and IR coding modes can achieve the same coding efficiency, whereas the intra coding mode, which allows the lowest delay, requires significantly higher data rates.

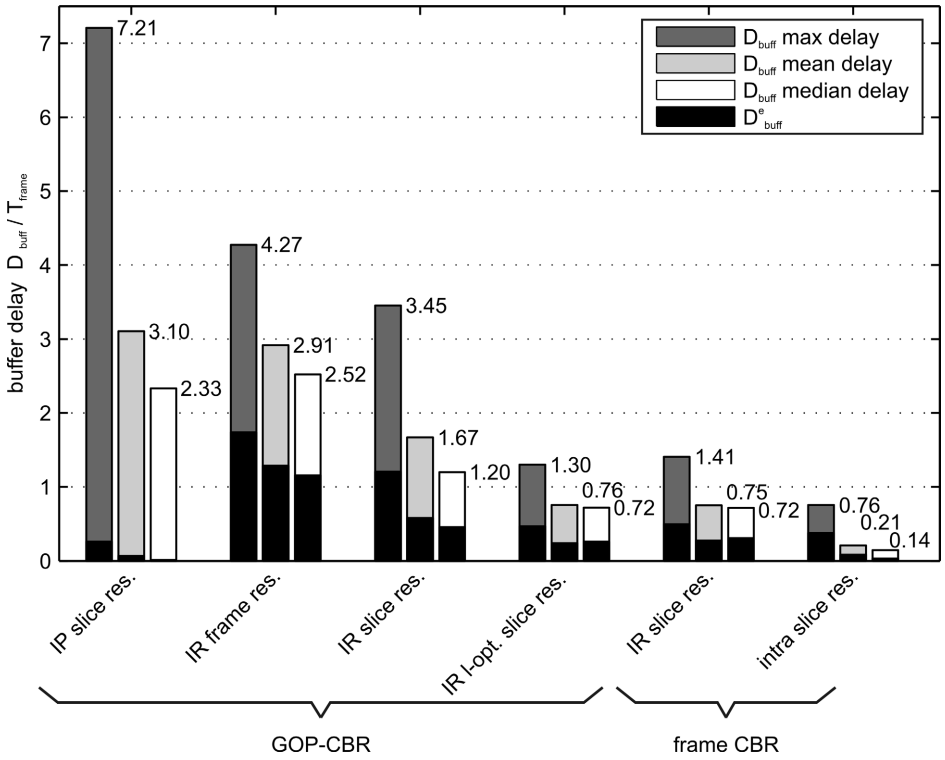


Figure 5.22: Coding mode buffer delay summary ($N_{GOP} = 12$ frames). The black bars indicate the contribution of the encoder buffer delay to the total buffer delay.

For the two IP coding mode simulation the maximum delay is above 7 frame periods, which is more than 2.3 times higher than the average and median delay values of this coding mode. As shown in figure 4.20 (page 89), slice level analysis allows an earlier start of transmission compared with the frame resolution analysis which results in a delay reduction of approximately 1.6 to 1.8 T_{frame}.

The three GOP-CBR IR coding simulations show a continuous delay reduction from frame resolution simulation to the latency-optimized slice resolution simulations. The maximum delay of frame resolution simulation IR is 4.27 T_{frame} which is approximately 170 ms for 25 Hz frame rate. In contrast to the IP coding mode, the

slice resolution analysis has a slightly lower potential of 0.8 to 1.3 T_{frame} delay reduction.

Compared with the IP coding mode, the IR slice resolution simulations show a delay reduction by 52 % for the maximum delay sequences. Assuming slice-level process-scheduling at 25 Hz frame rate, this means that the slice IR operation mode allows 138 ms delay compared with 288 ms buffer delay for IP-mode if the parameters are selected for measured worst-case sequences.

A further delay reduction down to 1.30 T_{frame} (52 ms) can be achieved if the intra refresh pattern associated with the lowest delay is selected. In this coding mode, the mean and median delay values are approximately 0.7 T_{frame} which is a factor of 69 % and 76 % below the median and mean delays of IP-mode slice resolution simulations. It can be concluded that the optimal intra refresh allows a delay reduction by at least 69 % compared with IP coding and 45 % compared with non-optimized IR.

The delay analysis clearly indicates that the frame-CBR intra refresh can achieve similar buffer delays as the latency-optimized GOP-CBR. As it requires about 15 % to 30 % higher data rates than the comparable GOP-CBR IR and IP coding modes it is only an interesting option if the l-opt. IR fails or cannot be used.

Extremely low delays can be realized with the intra coding mode at significantly reduced coding efficiency. For the slice resolution simulations, the optimal GOP-CBR intra refresh mode still results in 1.7 times higher maximum delays and 3.6 times higher mean delays than the intra coding mode. Similarly, the frame-CBR intra refresh increases the delays by a factor of 1.9. Hence, if delays significantly below one frame period are required, only the intra coding mode can be used and a careful tradeoff between delay adjustment and allowable data rate fluctuations must be found.

5.9.2 System latency summary

The system latencies for all considered coding modes are summarized in table 5.3. For the standard coding modes intra, IP and IPB a detailed discussion of the latencies is given in section 4.6. The three intra refresh coding modes have identical processing and capture delays as the intra and IP coding modes. Consequently, the system delays of these modes can be compared directly by their buffer delays.

Considering only the buffer delays it can be seen that the intra refresh modes smoothly fill the gap between the intra coding mode with extremely low latency and the IP coding mode with better coding efficiency but higher latency. For standard parameters $T_{frame} = 40\ ms$ and $T_{slice} = 0.025\ T_{frame} = 1\ ms$, the intra coding modes can be realized with system latencies below 33 ms. For applications which

require guaranteed system latencies below 100 *ms*, the latency-optimized intra refresh (D_{sys} = 55 *ms*) or frame-CBR intra refresh (D_{sys} = 60 *ms*) can be used. While the latency-optimized intra refresh allows superior coding efficiency if the joint data rate and latency optimization (section 5.8) is used, the frame-CBR intra refresh has slightly lower coding efficiency and smaller algorithmic complexity.

For applications which are not extremely delay sensitive, the standard frame-CBR intra refresh can be used without latency optimization. It allows system delays of 141 *ms* which is significantly below the 291 *ms* delay of the IP coding mode.

Table 5.3: Coding mode latency summary (N_{GOP} = 12 for all IP, IPB and IR modes).

Coding mode	CBR mode	D_{cap}	D^e_{proc}	$D_{buff} = D^e_{buff} + D^d_{buff}$		D^d_{proc}
				unconstrained CBR	*constrained CBR* (N_{GOP} = 12)	
Intra	frame	1 T_{slice}	1 T_{slice}	2 T_{frame}	0.21 .. 0.76 T_{frame}	1 T_{slice}
randomized Intra	block (slice)	1 T_{frame}	1 T_{slice}	2 T_{block}	0.02 .. 0.05 T_{frame}	1 T_{frame}
IP	GOP	1 T_{slice}	1 T_{slice}	1 T_{frame} + 1 T_{GOP}	3.10 .. 7.21 T_{frame}	1 T_{slice}
IPB	GOP	1 T_{slice}	1 T_{slice} + (1+N_B) T_{frame}	1 T_{frame} + 1 T_{GOP}	IBP: 5.31 .. 8.95 T_{frame} IBBP: 5.31 .. 9.06 T_{frame}	1 T_{slice}
IR frame-CBR	frame	1 T_{slice}	1 T_{slice}	2 T_{frame}	0.75 .. 1.41 T_{frame}	1 T_{slice}
IR GOP-CBR	GOP	1 T_{slice}	1 T_{slice}	2 T_{GOP}	1.67 .. 3.45 T_{frame}	1 T_{slice}
IR GOP-CBR l-opt	GOP	1 T_{slice}	1 T_{slice}	2 T_{GOP}	0.76 .. 1.30 T_{frame}	1 T_{slice}

5.9.3 Latency, data rate and error recovery comparison chart

Bringing all the previous results together, the factors data rate and error recovery time are inspected in more detail along with the system delay estimation. Based on these three parameters, this section gives the system designer a decision base for selecting the most appropriate coding mode and parameters for delay-critical applications. All delay simulations are based on optimal slice-level process-scheduling and slice resolution buffer analysis.

In figure 5.23 the delay range for the various coding modes is summarized with respect to the measured or estimated data rates based on the ten video sequences. The delay bars indicate the observed range of buffer delays between the measured *mean* and *maximum* buffer requirement values. The color indicator is located at the average of the measured mean and maximum buffer requirements.

As discussed in section 5.4, the intra refresh coding modes have an approximately two times higher recovery time than the IP and IPB coding modes. The exact maximum error recovery times for all simulations are listed in table 5.4 and the refresh groups and patterns for all recovery times are given in appendix E.

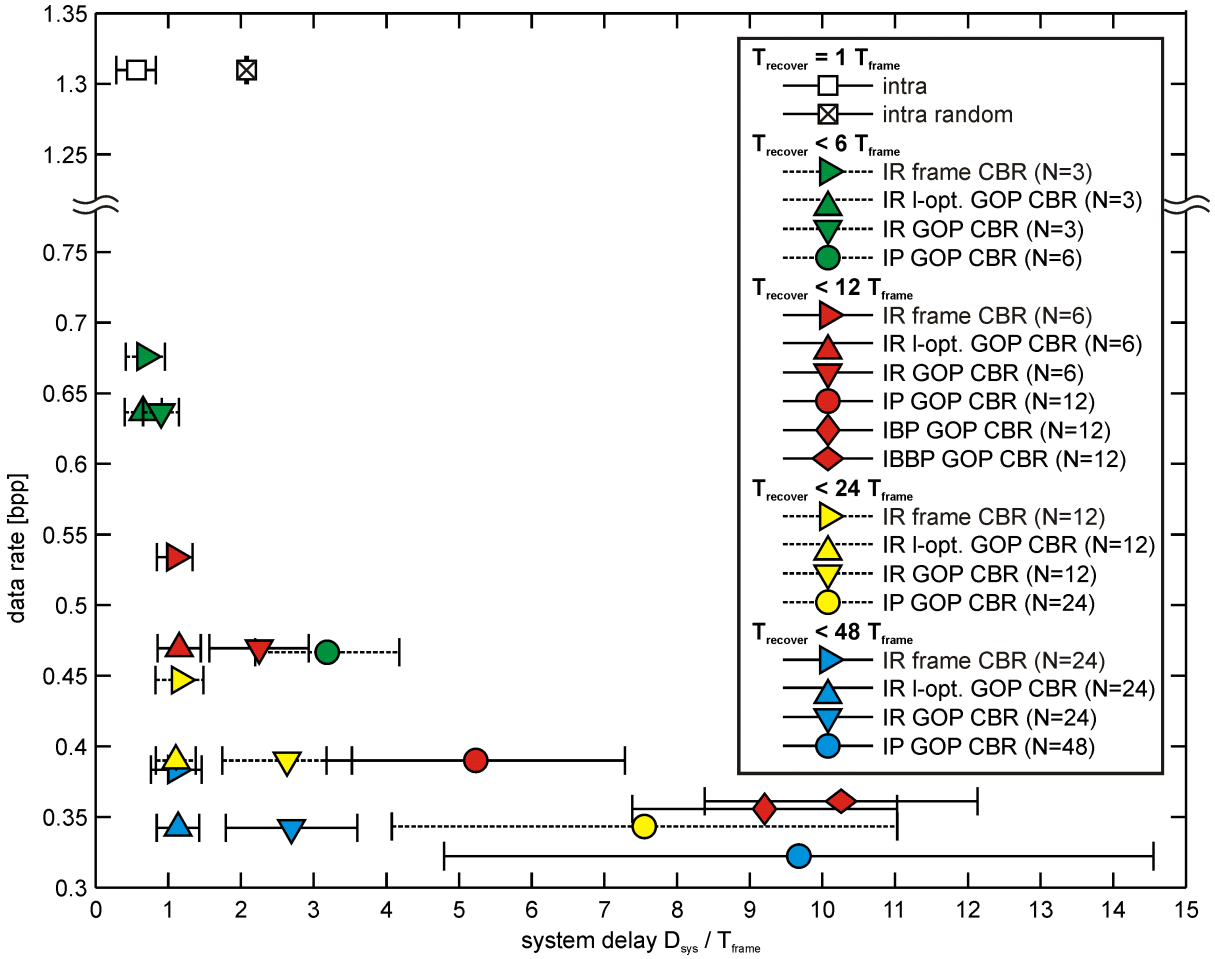


Figure 5.23: System delay vs. data rate and recovery time summary.

Table 5.4: Coding mode error recovery, bitrate and buffer delays.

Coding mode, GOP-size	$\frac{T_{recover}^{max}}{T_{frame}}$	$\frac{\text{mean bitrate}}{bpp}$	$\frac{\text{mean } (D_{buff})}{T_{frame}}$	$\frac{\max (D_{buff})}{T_{frame}}$
intra	1	1.31	0.21	0.76
intra random	1	1.31	0.03	0.07
IR frame-CBR ($N_{GOP} = 3$)	5	0.68	0.34	0.88
IR l-opt. GOP-CBR ($N_{GOP} = 3$)	5	0.64	0.32	0.84
IR std. GOP-CBR ($N_{GOP} = 3$)	5	0.64	0.58	1.07
IP GOP-CBR ($N_{GOP} = 6$)	6	0.47	2.12	4.10
IR frame-CBR ($N_{GOP} = 6$)	11	0.53	0.77	1.26
IR l-opt. GOP-CBR ($N_{GOP} = 6$)	11	0.47	0.78	1.37
IR std. GOP-CBR ($N_{GOP} = 6$)	11	0.47	1.49	2.86
IP GOP-CBR ($N_{GOP} = 12$)	12	0.39	3.10	7.21
IBP GOP-CBR ($N_{GOP} = 12$)	12	0.36	5.31	8.95
IBBP GOP-CBR ($N_{GOP} = 12$)	12	0.36	5.31	9.06
IR frame-CBR ($N_{GOP} = 12$)	23	0.45	0.75	1.41
IR l-opt. GOP-CBR ($N_{GOP} = 12$)	23	0.39	0.76	1.30
IR std. GOP-CBR ($N_{GOP} = 12$)	23	0.39	1.67	3.45
IP GOP-CBR ($N_{GOP} = 24$)	24	0.34	4.00	10.95
IR frame-CBR ($N_{GOP} = 24$)	47	0.38	0.68	1.38
IR l-opt. GOP-CBR ($N_{GOP} = 24$)	47	0.34	0.77	1.35
IR std. GOP-CBR ($N_{GOP} = 24$)	47	0.34	1.71	3.53
IP GOP-CBR ($N_{GOP} = 48$)	48	0.32	4.72	14.48

The results in figure 5.23 are sorted in four groups of different error recovery times, ranging from one frame to approximately one second recovery time for 25 *fps* video. From a comparison of different coding modes at different recovery times the following conclusions can be drawn:

1. The **intra coding** mode offers the shortest buffer delay of less than 0.75 T_{frame} and instantaneous error recovery.

2. The **GOP-CBR IR with latency optimization** offers the best coding efficiency with delays between 0.7 and 1.4 T_{frame}. With respect to IP-coding the delay is reduced by a factor of 5 to 10 but a 6 % to 35 % higher data rate is required to achieve the same recovery time.

3. Using the **frame-CBR IR** mode allows short delays below 1.4 T_{frame} with temporal prediction at the cost of 19 % to 44 % higher data rate compared with IP-coding with the same recovery time. Furthermore, the delay is reduced by a factor of 5 to 10 compared with IP-coding and the data rate overhead decreases with longer GOPs.

4. The **GOP-CBR IR without latency optimization** increases the delay by roughly a factor 2 to 2.6 compared with the GOP-CBR IR with latency optimization. The delay reduction with respect to IP-coding is a factor of 2.5 to 4.

5. The delay of the **IP-coding** modes increases roughly with a square root law if the recovery time is increased.

6. The delay of **all IR modes** is below $3.5\ T_{frame}$ for the standard mode and $1.4\ T_{frame}$ for the latency-optimized mode. There is no indication that it increases significantly for longer recovery times.

Based on the presented analysis and simulation results it can be concluded that the intra refresh method offers the highest coding efficiency for system latencies between $0.5\ T_{frame}$ and $2.2\ T_{frame}$. The simulations also clearly show that standard IP and IPB encoding cannot be used for low-latency transmission below $100\ ms$ system delay unless the maximum predictive coding gain is limited to guarantee lower delays. Especially in the case of H.264 video compression, which mostly benefits from improved temporal prediction tools, this would result in a significant drop of coding efficiency.

6 Implementation Considerations for Low-Latency Video Compression Systems

6.1 Introduction

In this chapter the implementation of a low-latency video encoder on a programmable DSP platform is described briefly [93]. The realization of a low-latency MPEG-2 encoder proves the feasibility of the process-scheduling optimizations described in chapter 4 on a commonly used architecture.

The target architecture, a Blackfin BF561 [83] dual-core DSP is shown in figure 6.1. The two identical processor cores have a modified Harvard architecture with separated instruction and data memory interfaces. The internal address and data registers as well as all memory interfaces are 32 bits wide. The processor cores have a reduced instruction set (RISC) with an 8 stage instruction pipeline and a maximum clock frequency of 600 *MHz*. Each processor core has two 16-bit multipliers and four 8-bit video ALUs (algorithmic-logic units) which are programmed by special SIMD (single instruction multiple data) instructions. The architecture enables parallel operation of many arithmetic operations with load/store operations. The program sequencer offers flow optimization by static branch prediction and zero overhead loops. Each processor has 68 *kbytes* of dedicated L1 data memory and 32 *kbytes* of L1 instruction memory. One half of the L1 (level-1) memories can be configured as either automatic cache or static memory. The L1 memories can be accessed at full core clock frequency if no bus collisions occur.

Both processor cores share a 128 *kbyte* L2 memory bank which operates at the core clock frequency. The external SDRAM operates with a maximum external clock domain frequency of 133 *MHz*. Several DMA (direct memory access) engines support the processor cores in transferring data from the external SDRAM to L1 and L2 (level-2) memories as well as L1 to L2 memory transfers. On transferring data the DMA engines can re-organize two-dimensional data arrays. DMA transfers are usually initiated by the processor and controlled by data structures. On the completion of a DMA transfer the processor can be notified by an interrupt.

Furthermore, the DSP has several parallel and serial interfaces which are most commonly operated by DMA transfers. The two parallel peripheral interfaces (PPI) offer flexible connection to high-speed A/D and D/A converters and support ITU-T BT.656 video input. For locking the processor on the video source it is also possible to generate interrupts by the vertical and horizontal synchronization signals of the video input signal.

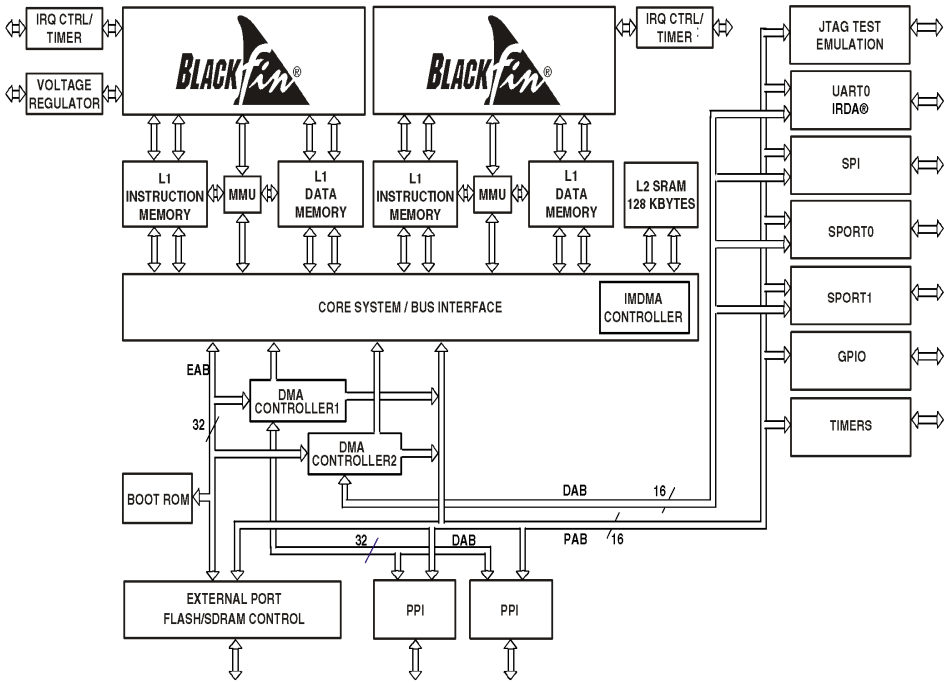


Figure 6.1: Blackfin multimedia DSP Architecture [84].

6.2 DSP implementation of an MPEG-2 encoder

6.2.1 Detailed functional overview

The major functional blocks of the MPEG-2 encoder were discussed in section 2.3.2. figure 6.2 gives a more detailed insight into the encoder architecture with all relevant functions for encoding progressive and interlaced video.

The video input was realized with the PPI port of the DSP and the incoming data is transferred to the L2 memory in sections of 16 lines. As the video input signal is sampled in 4:2:2 color format, the DSP must perform the separation of interleaved YUYV video data and subsampling of color components in software. In the case of an intra or IP-mode encoder, the frame reordering is not needed.

The synchronization of the internal processing and the input video signal is achieved by the controller, which monitors interrupts generated by the video interface. It is also responsible for the allocation of memories, and controls the memory-to-memory DMA transfers. As the reference frame has a size of 608 *kbytes*, it must be stored in the external memory and sections of the reference frame are transferred with DMA to the L2 memory bank (see also section 6.2.3).

On the completion of an video input DMA transfer, the controller starts the video encoder processing. The sequence of algorithmic operations is illustrated by the numbers in the block diagram. The multiplexing of DCT data and other stream data is performed on the fly at several stages of the processing.

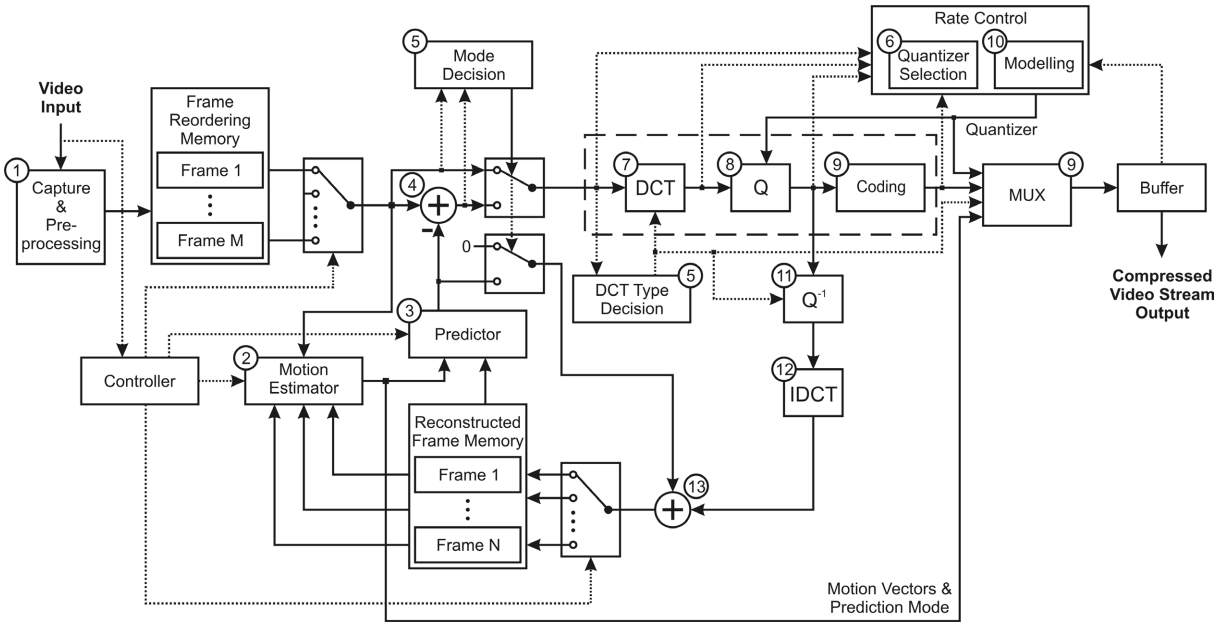


Figure 6.2: Detailed structure of an MPEG-2 video encoder.

If MPEG-2 transport stream encapsulation of the encoded video stream is required, the necessary headers and fields can be inserted directly in the multiplexer (MUX) stage of the encoder. This also has the advantage that all header data can be considered in the rate control, allowing optimal usage of the available channel bandwidth.

6.2.2 Program-flow optimizations for low-latency encoders

The fundamental video coding algorithm described in section 2.3 leaves the system designer certain options for organizing the sequence of operations on a macroblock level. Depending on the program flows illustrated in figure 6.3, different minimum latencies and memory requirements will result for a real-time video encoder. The illustrated examples for a multi-processor implementation were validated for the relatively simple MPEG-2 codec, which has only few data feedback dependencies. The possibilities for inserting pipelining stages generally depend on the data parallelism of the algorithm. In practical implementations a careful tradeoff between the pipelining performance increase and workload averaging, as well as memory requirements of the sub-units will be necessary.

In the standard configuration of current video standards, the macroblocks of a frame are coded in a strictly increasing order in lines of macroblocks starting at the top left corner of the frame. In the following discussion, a complete line of macroblocks is called a slice, although most video standards use a more flexible definition for this term.

In a frame-based processing scheme illustrated in figure 6.3 (a), the x and y macroblock loops are located inside the major processing functions. The sequence of processing in the original MPEG-2 TM5 codec is to perform the motion estimation for all macroblocks of a frame in a first step, then the prediction for all macroblocks and so on. In a straightforward real-time implementation, the encoder will capture the whole frame before the motion estimation processing begins. Assuming that the processing takes one frame period, a frame-based processing scheme introduces a latency of at least two frame periods in the encoder. For implementations on parallel architectures, pipelining stages could be inserted after each processing stage, introducing additional latencies of one frame period.

Moving the x and y loops up to the frame level as shown in figure 6.3 (b) minimizes the latency and memory requirements for storing intermediate results. Similar processing schemes are used by RD-optimized codecs like H.264, where several iterations for motion estimation, prediction, DCT and quantization are performed for each macroblock to determine the best coding mode. If a non-iterative program flow is assumed, pipelining stages can be inserted after each major processing stage in the codec. However, a macroblock processing scheme is disadvantageous for supporting parallel processing architectures like processors with dedicated function co-processors or multi-core processors, as the frequent synchronization of pipelined processing units increases the complexity and can lead to unbalanced workloads.

A good tradeoff between latency and complexity can be achieved by slice-based processing schemes. As illustrated in figure 6.3 (c), the x-loop is on the same level as in the frame processing scheme, whereas the y-loop is moved up to the frame level. In this approach, pipelining stages can be inserted after each x-loop operation, which eases the implementation on parallel processing architectures.

In the slice-based implementation, processing units must only be synchronized at the end of a slice, whereas the macroblock pipelining results in a significantly higher synchronization frequency (numbers given for SDTV resolution):

$$\text{macroblock pipelining: } f_{sync,MB} = 25\frac{frames}{s} \cdot 1620\frac{syncs}{frame} = 40500\frac{syncs}{s} \quad (6.1)$$

$$\text{slice pipelining: } f_{sync,slice} = 25\frac{frames}{s} \cdot 36\frac{syncs}{frame} = 900\frac{syncs}{s} \quad (6.2)$$

Reducing the frequency of synchronizations also allows more flexible distribution of workloads on processing units, which is especially critical in the entropy coding unit where the workload increases with the data rate. Considering figure 3.14 and figure 3.15, it becomes obvious that the amount of bits generated by a single macroblock varies significantly. Hence, in a macroblock-based processing system, each macroblock must be processed by the entropy coder within $1/40500$ seconds no matter how complex it is. As the average data rate is far below the peak

macroblock data rate, it makes sense to average-out these workload variations using a slice-based processing scheme.

Longer synchronization intervals generally allow better averaging of workload fluctuations but they also increase the system latency. For an I-frame-only video encoder, a fairly constant workload in the entropy coding unit can be achieved if the CBR interval is equal to the processing interval, e.g. slice-based processing with a constant slice bit rate.

Depending on the architecture and workloads, the slice-level pipelining stages can be rearranged. The combined MB/slice processing in figure 6.3 (d) was developed according to the profiling results of the DSP-optimized MPEG-2 encoder. For the dual-core DSP target architecture, the system is split into two pipelining stages between the x1 and x2-loop. Each loop is mapped to one DSP core which results in a balanced workload for the desired coding parameters.

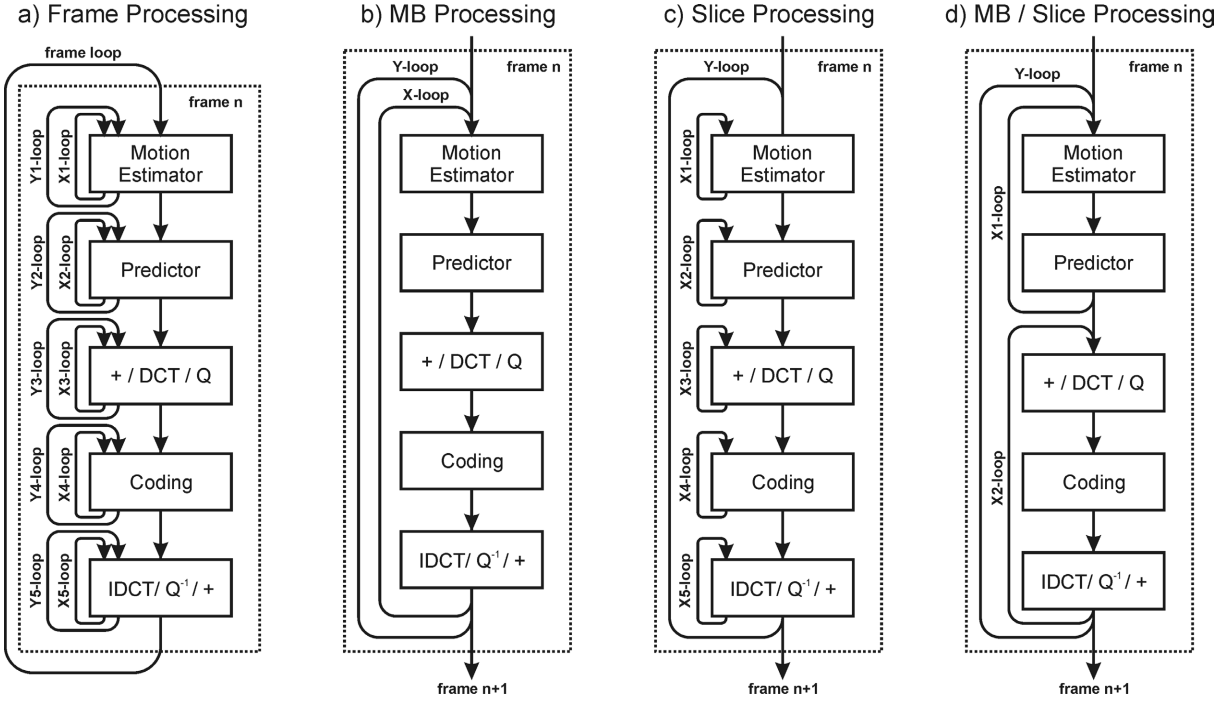


Figure 6.3: Program flow options for video encoders.

6.2.3 Memory usage and access optimizations

6.2.3.1 Frame memory requirements

High-performance signal- and general purpose processors use on-chip SRAM (static RAM) memories to reduce the latencies caused by accesses to external DRAM (dynamic RAM). The size of the on-chip memories is however a significant cost factor, as the static memory cells are rather large and expensive in terms of silicon area. Configuring the on-chip SRAM as cache allows mirroring fragments of

slower external RAM which are frequently accessed in fast internal RAMs. For optimal cache performance and execution speed, it is however necessary to localize data accesses for memory-intensive algorithms.

For the MPEG-2 encoder application, storing and reading the video frames is the most critical memory operation if the size of the frame data is significantly larger than the cache memory. A detailed memory organization diagram of the MPEG-2 reference encoder in IP coding mode is shown in figure 6.4.

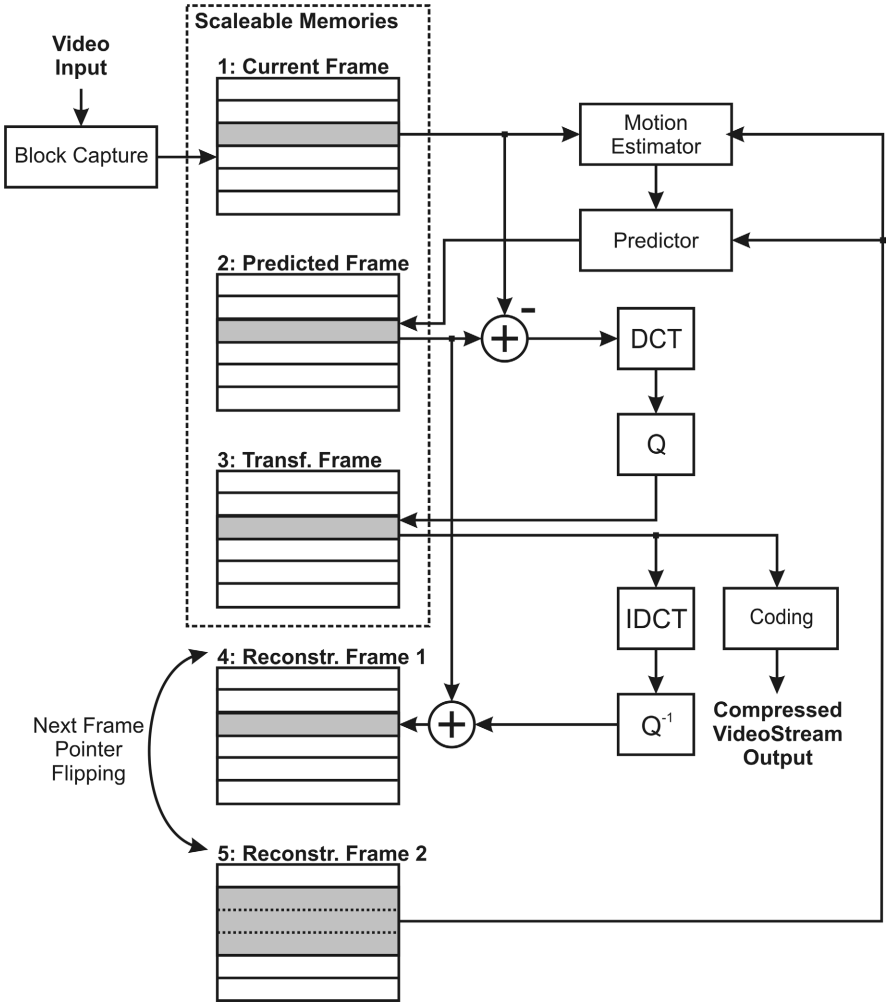


Figure 6.4: IP-mode video encoder frame memory organization.

The original implementation allocates three pixel memories for the current frame (1) and the reconstructed frames (4), (5). These memories correspond to figure 6.2 “Frame Reordering Memory” and “Reconstructed Frame Memory”. Furthermore, intermediate results of the prediction (2) and the DCT (3) are saved in frame memories because they are needed by more than one following processing blocks. For SDTV resolution, the pixel memories 1, 2, 4 and 5 each occupy $720 \cdot 576 \cdot 1.5\ bytes = 622\ kbytes$ for 4:2:0 color sub-sampling. As the intermediate results of the DCT have 10 bits precision, frame memory 3 usually consumes twice the size of the eight bit pixel memories. The frame memories 1 to 3 can be re-used

instantaneously, whereas the reconstructed frame memories 4 and 5 are accessed at half frame frequency. On the completion of a frame in IP-coding mode, the reconstructed frame becomes the reference for the next frame and the previous reference frame is not needed any more and can be reused for the reconstructed frame memory. Swapping the two frames can be achieved by swapping the pointers to the memory sections.

The original TM5 reference software uses a **frame-based processing** scheme, i.e. first the motion estimation for all macroblocks of a frame is performed, then prediction for all MBs and so forth. As each processing step must access the whole image from external memory, this processing sequence results in very high memory bandwidth requirements and bad cache usage.

For efficient cache usage it is favorable to localize memory data accesses by processing only sections of the image which fit into the cache. The lower-bound for localization is the **macroblock-based processing** on 16 by 16 pixel image regions. This allows reducing the cached parts of frame memories 1, 2 and 3 to 16 by 16 pixels which easily fit into CPU caches. From the old reference frames, only the pixels of the search window around the current macroblock are accessed, e.g. a 48 by 48 pixel image segment for the chosen implementation. The color components of the reference frame are less critical, as they are only needed once in the temporal prediction routines.

On the implementation side, the block-based processing requires an adaptation of the frame memory address calculations in low-level functions because of the modified picture geometry in the cache. Furthermore, splitting the coding task for efficient usage of current multi-processor architectures becomes more difficult because of the high amount of communication needed for synchronizing different processes on a macroblock level.

The **slice-based processing** overcomes the drawbacks of the block-based processing by caching complete rows of macroblocks. In combination with the virtual array indexing scheme described in the next section, the modification of frame memory address calculations can be avoided, thus resulting in easier system implementation and verification. Also, the communication overhead for multi-processor implementations becomes more reasonable, as the synchronization interval is reduced from 1620 Macroblocks per frame (40.5 *kHz*) to 36 slices per frame (900 *Hz*) for SDTV resolution.

Finally, combining the slice-based and block-based processing schemes allows reduction of the transformed frame memory while maintaining the implementation advantages of the slice-based approach. In the case of the TM5 reference software, the transformed frame is organized as a multi-dimensional structure which can be easily adapted for frame or block-based addressing. For the target

architecture, slice-based processing with a cached image size of one slice for the current frame and three slices for the reference frame was chosen.

6.2.3.2 Motion estimation memory

Optimizing the memory requirements of the motion estimation and prediction functions of the video encoder is critical due to the frequent accesses and large size of the referenced image section. As illustrated in figure 6.4, accesses to the current frame, predicted frame, transformed frame and reconstructed frame are usually restricted to a small, 16 line region of a frame. For motion estimation it is however necessary to cover a sufficiently large vertical search range which is usually at least +/- 16 lines around the current slice. Hence, even for a relatively small vertical search range it is necessary to provide three rows of macroblocks from every reference frame in the internal RAM. In the following discussions, border effects which result in a slight reduction of memory copy operations are not considered.

Figure 6.5 shows different memory organization scenarios for the motion estimation. Considering macroblock 9 in line one of the current frame and a search range of +/- 16 pixels in both image dimensions, pixel data from macroblocks 0, 1, 2, 8, 9, 10, 16, 17 and 18 is needed for reference. The **macroblock search window** results in the smallest memory usage but at the same time it requires the highest number of load and copy operations. After the motion estimation for the current macroblock is performed, the content of the search window is copied one column to the left, resulting in $2 \cdot 3 \cdot N$ cache copy operations for a whole frame of N pixels. Furthermore, the new content of the right most column is loaded from external memory, resulting in $3 \cdot N$ load operations.

In the case of a slice-size search window, macroblock lines 0, 1 and 2 must be buffered, requiring $48 \cdot 720 \cdot 1.5$ bytes = 51.8 *kbytes* memory for SDTV resolution.

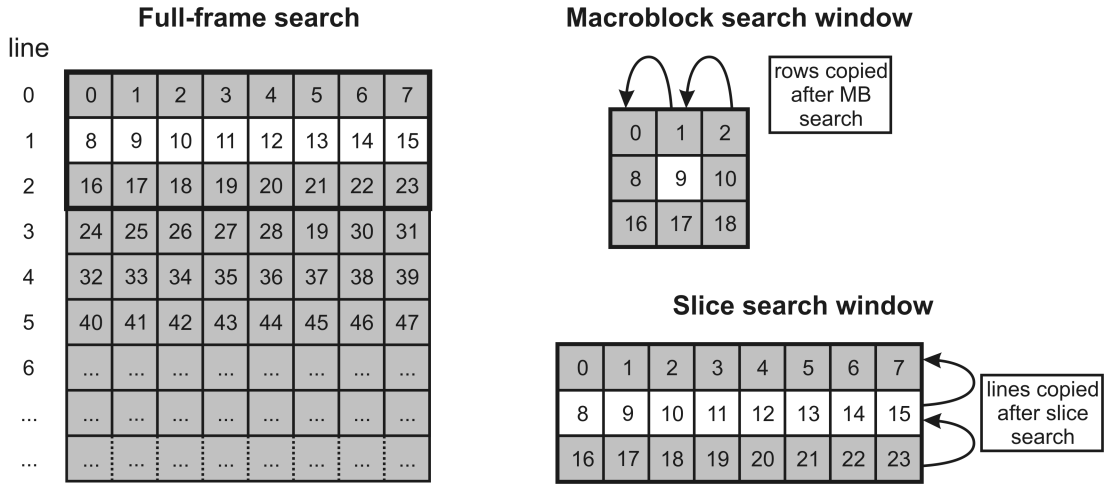


Figure 6.5: Memory organization for windowed motion estimation.

Using the **slice search window** approach, it is possible to shift the memory content in the frame segment after the motion estimation was performed for all macroblocks of a row. Therefore, the N pixels of the reference frame are only copied once from the external memory to the internal RAM which saves memory access bandwidth and energy. On the other hand it is necessary to copy $2 \cdot N$ pixels for the complete frame motion estimation inside the cache. If a simple slice-by-slice copying is implemented, the motion estimation must be stopped during the copy operation. It is however possible to perform the copying on columns of macroblocks from left to right as soon as the regions are not any more needed as reference.

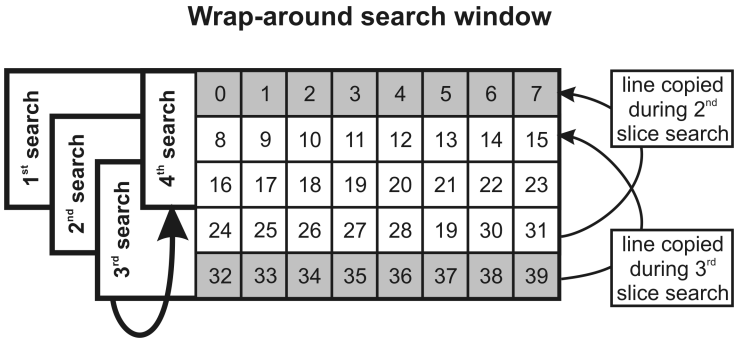


Figure 6.6: Wrap-around search window for motion estimation with reduced memory copy operations.

With the wrap-around search window, the copy operations can be further reduced to $2/3 \cdot N$ and can be executed in parallel with the motion estimation. Therefore, depending on the size of the available cache memories it is possible to exchange memory requirements and copy operations according to the resources of the architecture.

Table 6.1: Search window reload and move operations.

Memory size	cache memory	load operations	copy operations
frame	$1\ N$	$1\ N$	0
macroblock	$9 \cdot 256$	$3\ N$	$6\ N$
slice	$3\ N/M$	$1\ N$	$2\ N$
wrap-around	$5\ N/M$	$1\ N$	$2/3\ N$

N = number of pixels per frame, M = slices per frame

7 SUMMARY & CONCLUSIONS

In the course of this thesis an in depth analysis of low-latency video compression systems was performed. Under the constraint of CBR operation, all relevant coding modes were analyzed with respect to the algorithmic and buffer latencies as well as the overall coding efficiency and resynchronization intervals. The main outcomes of this thesis are the modified intra refresh strategy and the two optimization algorithms which enable low-latency video transmission with the full efficiency of comparable predictive coding methods.

For achieving low buffer delays at CBR operation it is generally advantageous to generate a constant amount of data in a short time interval with minimum intrusion of the rate control algorithm. The discussion in this document showed that this goal is contradictory to the concept of video coding with temporal prediction which is the key technology for high-efficiency video coding. Therefore, in the course of this thesis special attention was paid to the intra refresh coding method which enables high-efficiency temporal prediction in low-latency video transmission.

In the initial coding mode analysis the temporal data rate variations in the encoded video stream were examined for the MPEG-2 and H.264 video standards. Based on the observation of temporal rate fluctuations an inherent CBR interval was defined for each coding mode. For the intra-frame compression mode, the sequence and sub frame-level analysis clearly shows that the data rate is directly related to the complexity of the image texture. The frame-level analysis for the IP compression mode indicates that the compression ratio of predicted frames with respect to intracoded frames has a large range of 50 % up to 90 %.

The subsequent basic latency analysis introduced a system model consisting of a video encoder, CBR transmission channel and a video decoder. According to this model the process-scheduling for an optimized implementation was investigated for the different coding modes. It was shown that the lower bound delay of intra-coded transmission is defined by the conventional frame-resolution buffer modeling resulting in at least two frame periods delay. In the case of predictive coding methods the system delay is also dominated by the buffer delay which increases with the temporal prediction gain. Therefore, IP and IPB-coding methods introduce more than half a second of delay to achieve the full efficiency of predictive coding.

In the next step the latency analysis was refined with simulation data of ten video sequences using the leaky bucket model. As the traditional frame-level modeling of the encoder buffer does not allow buffer delays below two frame periods, the buffer analysis was performed with sub-frame resolution. The described

methodology enables calculation of buffer requirements based on spatial data rate distributions of I and P-frames and the applied coding mode.

In order to overcome the high delay requirements of IP-coding, a region-based intra refresh was introduced which embeds sections of intra-coded macroblocks in predicted frames. In this way the large I-frames can be avoided and the frame-level data rate variations are significantly reduced. In a first approximation it was assumed that the intra refresh method achieves a constant data rate per frame which results in similar buffer delay requirements as for the intra coding mode. Due to the sequential transmission of intra information, the guaranteed error recovery time of IR data streams is approximately doubled with respect to IP data streams.

A detailed analysis showed that the intra coding mode can achieve system delays below one frame period if coding efficiency is not a major concern. In addition to this it was shown that only the intra refresh method enables buffer delays below two frame periods in combination with high-efficiency temporal prediction. While the frame-CBR intra refresh enables delays of less than 1.4 frame periods, it requires up to 30 % higher data rate than comparable IP coding methods. Yet, this coding mode can reduce the data rate by at least 50 % compared with intra coding. If higher efficiency is required, the GOP-CBR intra refresh can be employed, which enables buffer delays in the range of 3.5 frame periods with maximum efficiency. It is shown that it is possible to reduce the buffer latency of the GOP-CBR mode to 1.3 frame periods by selecting a refresh strategy which minimizes the buffer requirements.

As the intra refresh method is an enabling technology for low delay video transmission, a second focus of this thesis lies in the optimization of this technique. The described intra refresh method groups adjacent macroblocks into rectangular regions and transmits one region in intra mode in each frame. In order to avoid back-propagation of previous transmission errors the intra refresh requires a border protection mechanism. Hence, at the border of refreshed regions the motion search range must be restricted, which can lead to a loss in coding efficiency. In order to overcome this drawback, an algorithm was developed which adapts the intra refresh to the global motion by choosing an optimal strategy out of a predefined set of refresh strategies. As the chosen strategy also has an impact on the buffer requirements, a joint optimization for latency and coding efficiency is proposed. With a combination of these optimization algorithms the intra refresh can guarantee a buffer delay of 1.4 frame periods with maximum temporal prediction coding efficiency.

8 BIBLIOGRAPHY

[1] S. R. Ely, "*MPEG video coding: A simple introduction*", European Broadcasting Union, EBU Technical Review Winter 1995, [online] www.ebu.ch/trev_266-ely.pdf.

[2] F. Schaeffel, "*Processing of Information in the Human Visual System*", Handbook of Machine Vision, Alexander Hornberg (Ed.), Wiley-VCH Verlag GmbH & Co, Weinheim, Germany, 2006, ISBN: 3-527-40584-7.

[3] J. Anderson, B. Anderson, "*The Myth of Persistence of Vision Revisited*", Journal of Film and Video, Vol. 45, No. 1, Spring 1993, pp. 3-12.

[4] S. Sidebottom, "*Effects of Illumination and Viewing Angle on the Modeling of Flicker Perception in CRT Displays*", Virginia Polytechnic Institute and State University, Industrial and Systems Engineering Department, Blacksburg, VA, USA, April 1997.

[5] Wikipedia, "*Persistence of vision*", [online] http://en.wikipedia.org/wiki/Persistence_of_vision, 2006.

[6] M. J. Nadenau, S. Winkler, D. Alleysson, M. Kunt, "*Human Vision Models for Perceptually Optimized Image Processing … A Review*", [online] http://dewww.epfl.ch/~nadenau/Research/Paper/HVS_Review_Final.pdf.

[7] M. Nadenau, "*Integration of human color vision models into high quality image compression*", Ph.D. Thesis No 2296, Signal Processing Laboratory, Swiss Federal Institute of Technology, Lausanne, Switzerland, Nov 2000.

[8] J. A. Ferwerda, S. N. Pattanaik, P. Shirley, D. P. Greenberg, "*A Model of Visual Adaptation for Realistic Image Synthesis*", Proceedings of SIGGRAPH'96, New Orleans, LA, USA, 4-9 August 1996, pp. 249-258.

[9] M. Trentacoste, "*Photometric Image Processing for High Dynamic Range Displays*", Faculty of graduate studies (Computer Science), University of Britisch Columbia, Canada, Jan. 2006.

[10] "*Digital Imaging and Communications in Medicine (DICOM), Part 14: Grayscale Standard Display Function*", National Electrical Manufacturers Association, 1300N. 17th Street, Rosslyn, Virginia 22209 USA, 2003.

[11] D. Williams, N. Sekiguchi, D. Brainard, "*Color, contrast sensitivity, and the cone mosaic*", Proc. Natl. Acad. Sci. USA, Vol. 90(21), Nov. 1993, pp. 9770-9777.

[12] H. Seetzen, W. Heidrich, W. Stuerzlinger, G. Ward, L. Whitehead, M. Trentacoste, A. Ghosh, A. Vorozcovs, "*High Dynamic Range Display Systems*", Proc.

of SIGGRAPH '04, ACM Transactions on Graphics, Vol. 23, No 3, August 2004, pp. 760-768.

[13] R. Mantiuk, G. Krawczyk, K.l Myszkowski, H.-P. Seidel, "*Perception-motivated High Dynamic Range Video Encoding*", Proc. of SIGGRAPH '04, ACM Transactions on Graphics, Vol. 23, No. 3, August 2004, pp. 733-741.

[14] H. A. Peterson, A. J. Ahumada, Jr., A. B. Watson, "*The visibility of DCT quantization noise*", Society for Information Display International Symposium Digest of Technical Papers, Vol. 24, 1993, pp. 942-945.

[15] E. Peli, "*Contrast sensitivity function and image discrimination*", Journal of the Optical Society of America A: Optics, Image Science, and Vision, Vol. 18, Issue 2, Feb. 2001, pp. 283-293.

[16] C. Reader, "*History of MPEG Video Compression … Ver. 4.0*", ISO/IEC JTC1/SC29/WG11, Document JVT-E066, Geneva, Switzerland, 9.-17. Oct. 2002.

[17] C. Christopoulos, A. Skodras, T. Ebrahimi, "*The JPEG2000 still image coding stystem: an overview*", IEEE Transaction on Consumer Electronics, Vol. 46, No. 4, Nov. 2000, pp. 1103-1127.

[18] International Telecommunication Union, ITU, "*International Standard ISO/IEC 13818-2, Information technology … Generic coding of moving pictures and associated audio information: Video*", First edition, ISO/IEC Copyright Office, Geneva, Switzerland, 1996.

[19] R. Wintner, "*Bits on the Big Screen*", IEEE Spectrum, Dec. 2006, pp. 37-42.

[20] ISO-IEC/JTC1/SC29/WG11, "*Test Model 5*", Coding of Moving Pictures and Associated Audio, 1994.

[21] "*H.264 JM Ref. Software*", [online] http://iphome.hhi.de/suehring/tml/download/.

[22] S. Borman, R. L. Stevenson, "*Image Sequence Processing*", Encyclopedia of Optical Engineering, Ed. Marcel Dekker, New York, NY, USA, 2002, pp. 840-879.

[23] M. Domansik, "*Image and video compression: current trends and perspectives*", International Conference on Signals and Electronic Systems, ICSES 2001, Lodz, Poland, 18-21 Sep. 2001.

[24] O. Egger, P. Fleury, T. Ebrahimi, M. Kunt, "*High-performance compression of visual information - A tutorial review - Part I: still pictures*", Proceedings of the IEEE, Vol. 87, No. 6, June 1999, pp. 976-1011.

[25] Sony Corporation, "*DVCAM Format Overview*", 2000, [online] http://www.sony.ca/dvcam/pdfs/dvcam%20format%20overview.pdf.

[26] International Telecommunication Union, ITU, "*Advanced video coding for generic audiovisual services, ITU-T Recommendation H.264*", ITU, Geneva, Switzerland, May 2003.

[27] L. Fan, S. Ma, Feng Wu, "*Overview of AVS video standard*", IEEE International Conference on Multimedia and Expo, ICME 2004, Taipei, Taiwan, 27-30 June 2004, Vol. 1, pp 423-426.

[28] S. Srinivasan, P. Hsu, T. Holcomb, K. Mukerjee, S. L. Regunathan, B. Lin, J. Liang, M.-C. Lee, J. Ribas-Corbera, "*Windows Media Video 9: overview and applications*", Signal Processing: Image Communication, Vol. 19, Issue 9, Oct. 2004, pp. 851-875.

[29] P. M. Kuhn, Georg Diebel, S. Herrmann, A. Kein, H. Mooshofer, A. Kaup, R. Mayer, W. Stechele, "*Complexity and PSNR-Comparison of several Fast Motion Estimation Algorithms for MPEG-4*", SPIE, Applications of digital image processing XXI, San Diego, CA, USA, 21-24 July 1998, Vol. 3460, pp. 486-499.

[30] T. Wiegand, G. J. Sullivan, G. Bjontegaard, A. Luthra, "*Overview of the H.264/AVC video coding standard*", IEEE Transactions on Circuits and Systems for Video Technology, Vol. 13, Issue 7, July 2003, pp. 560-576.

[31] L. Fan, S. Ma, Feng Wu, "*Overview of AVS video standard* ", IEEE International Conference on Multimedia and Expo, ICME 2004, Taipei, Taiwan, 27-30 June 2004, Vol.1, pp. 423-426.

[32] G. J. Sullivan, T. Wiegand, "*Video Compression - From Concepts to the H.264/AVC Standard*", Proceedings of the IEEE, Jan. 2005, Vol. 93, No. 1, pp. 18-31.

[33] C. A. Gonzales, H. Yeo, C. J. Kuo, "*Requirements for motion-estimation search range in MPEG-2 coded video*", IBM J. Res. Develop., Vol. 43, No. 4, July 1999, pp. 453-470.

[34] I.-H. Lee, R.-H. Park, "*A fast block matching algorithm using integral projections*", Proceedings Tencon '87 Conference, New York, NY, USA, Aug. 1987, Vol. 2, pp. 590-594.

[35] J.-S. Kim, R.-H. Park, "*A Fast Feature-Based Block Matching Algorithm Using Integral Projections*", IEEE Journal on Selected Areas in Communications, Vol. 10, No. 5, June 1992, pp. 968-971.

[36] K. P. Lim, S. Wu, D. J. Wu, S. Rahardja, X. Lin, F. Pan, Z. G. Li, "*Fast INTER Mode Selection*", ISO/IEC JTC1/SC29/WG11, Document JVT-I020, San Diego, CA, USA, 2-5 Sep. 2003.

[37] R. Braspenning, G. de Haan, "*Efficient motion estimation with content-adaptive resolution*", Proceedings. Int. Conference on Condsumer Electronics, ISCE 2002, Illmenau, Germany, 24-26 Sep. 2002, pp. E29-E34.

[38] A. Joch, F. Kossentini, "*Performance Analysis of H.26L Coding Features*", ITU-T/SG 16/VCEG, Document VCEGO42, 15th Meeting, Pattaya, Thailand, Dec. 2001.

[39] J. Ribas-Corbera, D. L. Neuhoff, "*Optimizing motion-vector accuracy in block-based video coding*", IEEE Transactions on Circuits and Systems for Video Technology, Vol. 11, No. 4, April 2001, pp. 497-511.

[40] Y. Liu, S. Oraintara, "*Fractional-pel motion refinement based on hierarchical adjustable dual-parobola model*", International Symposium on Communications and Information Technologies 2004, ISCIT 2004, Sapporo, Japan, 26-29 Oct. 2004.

[41] K. Jack, "*Video Demystified: a handbook for the digital engineer*", 3rd Ed., LLH Technology Publishing, Eagle Rock, VA, USA, ISBN 1-878707-56-6, 2001.

[42] B. Yan, K. W. Ng, "*An efficient error detection technique for MPEG-4 video streams*", First IEEE Consumer Communications and Networking Conference, 2004, CCNC 2004, Las Vegas, NV, USA, 5-8 Jan. 2004, pp. 442-446 .

[43] R. Talluri, "*Error-resilient Video Coding In The ISO MPEG-4 Standard*", IEEE Communications Magazine, Vol. 36, No. 6, June 1998, pp. 112-119.

[44] T. Stockhammer, M. M. Hannuksela, T. Wiegand , "*H.264/AVC in wireless environments*", IEEE Transactions on Circuits and Systems for Video Technology, Vol. 13, No. 7, July 2003, pp. 657-673.

[45] Z. Wang, A. Conrad Bovik, Hamid Rahim Sheikh, Ero P. Simoncelli, "*Image quality assessment: from error visibility to structural similarity*", IEEE Transactions on Image Processing, Vol. 13, No. 4, April 2004, pp. 600-612.

[46] Z. Wang, E. P. Simoncelli, "*An adaptive linear system framework for image distortion analysis*", Proceedings. IEEE International Conference on Image Processing, ICIP, Genoa, Italy, 11-14 Sep. 2005, Vol. 3, pp. III 1160-1163 .

[47] O. Rose, "*Statistical properties of MPEG video traffic and their impact on traffic modeling in ATM systems*", Report No. 101, University of Würzburg, Institute of Computer Science, Research Report Series, Würzburg, Germany, Febr. 1995.

[48] M. Baldi, Y. Ofek, "*End-to-end delay analysis of videoconferencing over packet-switched networks*", IEEE/ACM Transactions on Networking, Vol. 8, No. 4, Aug. 2000, pp. 479-492.

[49] A. Ortega, "*Variable bit-rate video coding*", in Compressed Video over Networks, M.-T. Sun and A. R. Reibman, Eds, Marcel Dekker, New York, NY, USA, 2000, pp. 343-382.

[50] H. Yasuura, "*System Design Methods for Low Energy Consumption Embedded Systems*", Summer School Electronic Media and Systems on a Chip, Dortmund, Germany, 19 Sept. 2000,.

[51] "*IEEE Standard for a High-Performance Serial Bus 1394-2008*", Microprocessor and Microcomputer Standards Committee of the IEEE Computer Society, USA, ISBN 978-0-7381-5771-9, 2008.

[52] MOST Cooperation, "*MOST Specification*", Rev. 2.5, June 2006, [pdf, online] http://www.mostcooperation.com.

[53] M. M. Hannuksela, Y.-K. Wang, M. Gabbouj, "*Isolated regions in video coding*", IEEE Transactions on Multimedia, Vol. 6, No. 2, April 2004, pp. 259-267.

[54] Y.-K. Wang, M. M. Hannuksela, "*Gradual Decoder Refresh Using Isolated Regions*", ISO/IEC JTC1/SC29/WG11, Document JVT-C074, Fairfax, VA, USA, 6.-10. May 2002.

[55] International Telecommunications Union, ITU, "*Video coding for low bit rate communications, ITU-T Recommendation H.263*", ITU, Geneva, Switzerland, Feb. 1998.

[56] J. Ribas-Cobera, P. A. Chou, S. L. Regunathan, "*A generalized hypothetical reference decoder for H.264/AVC*", IEEE Transactions on Circuits and Systems for Video Technology, Vol. 13, No. 7, July 2003, pp. 674-687 .

[57] Y. Wang, M. D. Srinath, "*Robust video coding based on tree structure motion compensation*", Ninth DSP Workship & First Signal Processing Education Workshop, Hunt, TX, USA, 15-18 Oct. 2000, [pdf, online] http://spib.rice.edu/DSP2000/.

[58] N. Färber, E. Steinbach, B. Girod, "*Robust H.263 compatible transmission for mobile video server access*", First International Workshop on Wireless Image/Video Communications Proc., IEEE, 1996, Loughborough, UK, 4-5 Sep. 1996, pp. 8 – 13.

[59] S.Aramvith, C.-W. Lin, S. Roy, M.-T. Sun, "*Wireless video transport using conditional retransmission and low-delay interleaving*", IEEE Transactions on Circuits and Systems for Video Technology, Vol. 12, No. 6, June 2002, pp. 558-565 .

[60] S. Aramvith, I-Ming Pao, Ming-Ting Sun, "*A rate-control scheme for video transport over wireless channels*", IEEE Transactions on Circuits and Systems for Video Technology, Vol. 11, No. 5, May 2001.

[61] J.-T. Wang, P.-C. Chang, "*Error-propagation prevention technique for real-time video transmission over ATM Networks*", IEEE Trans. on Circuits and Systems for Video Technology, Vol. 9, No. 3, April 1999, pp. 513-523.

[62] E. Krause, W. H. Paik, V. C. Liu, J. A. Heller, P. Shen, J. N. Esserman, "*Method and apparatus for refreshing motion compensated sequential video images*", U.S. Patent 5057916, United States Patent Office, USA, 15 Nov. 1991.

[63] T. D. Tran, L.-K. Liu, P. H. Westerink, "*Low-delay MPEG-2 video coding*", SPIE - International Society Optical Engineering, USA: SPIE, Vol. 3309, 1997, pp. 510-516.

[64] A. Shikakura, M. Kato, "*Image encoding apparatus having an intrapicture encoding mode and interpicture encoding mode*", U.S. Patent 5805223, United States Patent Office, USA, Sept. 8 1998.

[65] N. Färber, K. Stuhlmüller, B. Girod, "*Analysis of error propagation in hybrid video coding with application to error resilience*", Proceedings. International Conference on Image Processing, 1999, ICIP 99, Vol. 2, Kobe, Japan, 24-28 Oct. 1999, pp. 550-554.

[66] K. Stuhlmüller, N. Färber, B. Girod, "*Adaptive optimal intra update for lossy video transmission*", Proc. SPIE Vol. 4067, Visual Communications and Image Processing 2000, Vol. 4067, pp. 286-295.

[67] J. Kwon, J.-M. Jo, "*Low-delay encoding methods of moving pictures*", U.S. Patent 5745180, United States Patent Office, USA, 28 Apr. 1998.

[68] Q.-F. Zhu, L. Kerofsky, " *Joint source coding, tranport processing, and error concealment for H.323-based packet video*", IS&T/SPIE Conference on Visual Communications and Image Processing '99, San Jose, California, January 1999, SPIE Vol. 3653, pp. 52-62.

[69] Y. Watanabe, H. Aono, "*Moving image coding method and apparatus, and moving image coding and decoding system*", U.S. Patent 6668016, United States Patent Office, USA, Dec. 2003.

[70] J. Y. Liao, J. Villasenor, "*Adaptive Intra Update for Video Coding over Noisy Channels*", Proceedings. International Conference on Image Processing, New York, NY, USA, Vol. 3, 1996, pp. 763-6.

[71] J. Y. Liao, J. Villasenor, "*Adaptive intra block update for robust transmission of H.263*", IEEE Transactions on Circuits and Systems for Video Technology, Vol. 10, No. 1, Feb. 2000, pp. 30-35.

[72] M. H. Willebeek-LeMair, Z. -Y. Shae, Y. -C. Chang, "*Robust H.263 Video Coding For Transmission over the internet*", Proceedings. Seventeenth Annual Joint

Conference of the IEEE Computer and Communications Societies, INFOCOM '98, Vol. 1, San Francisco, CA, USA, 29 March – 2 April 1998, pp. 225-232.

[73] ISO/IEC JTC1/SC29/WG11/N2202, " Information Technology - Coding of Audio-Visual Objects: Visual, ", ISO/IEC 14496-2, ISO/IEC Copyright Office, Geneva, Switzerland, March 1998.

[74] S. T. Worrall, A. H. Sadka, A. M. Kondoz, P. Sweeney, "Motion adaptive intra refresh for MPEG-4", Electronics Letters, Vol. 36, Issue 23, 9 Nov. 2000, pp. 1924-1925.

[75] Y. Matsumura, T. Nakai, "Picture coding method and picture coder", U.S. Patent 6125144, United States Patent Office, USA, Sept. 26, 2000.

[76] A. Kurobe, S. Masaki, "Video coding apparatus, video coding method and storage medium containing video coding program", U.S. Patent 6333948, United States Patent Office, USA, 25 Dec. 2001.

[77] J. Ribas-Corbera, S. Lei, "Rate control in DCT video coding for low-delay communications", IEEE Transactions on Circuits and Systems for Video Technology, Vol. 9, No. 1, February 1999, pp. 172-185.

[78] U. Bayazit, "Macroblock data classification and nonlinear bit count estimation for low delay H.263 rate control", Proceedings of the International Conference on Image Processing, 1999. ICIP 99, Kobe, Japan, 24-28 Oct 1999, pp. 263-297.

[79] Z. He, Y. Kwan Kim, S. K. Mitra, "Low-delay rate control for dct video coding via rho-domain source modeling", IEEE Tr. on Circuits and Systems for Video Technology, Vol. 11, No. 8, August 2001, pp. 928-940.

[80] J.-C. Tsai, "Rate control for low-delay video using a dynamic rate table", IEEE Transactions on Circuits and Systems for Video Technology, Vol. 15, No. 1, Jan. 2005, pp 133-137.

[81] Y. K. Wang, M. M. Hannuksela, "Isolated Regions: Motivation, Problems, and Solutions", ISO/IEC JTC1/SC29/WG11, Document JVT-C072, Fairfax, VA, USA, 6-10 May 2002.

[82] M. M. Hannuksela, Y.-K. Wang, M. Horowitz, "Enhancements to FMO", ISO/IEC JTC1/SC29/WG11, Document JVT-D095, Klagenfurt, Austria, 22-26 July 2002.

[83] R. K. Kolagotla, J. Fridman, B. C. Aldrich, et. al., "High performance dual-MAC DSP architecture", IEEE Signal Processing Magazine, July 2002, pp. 42-53.

[84] Analog Devices Inc., :"ADSP-BF561 Blackfin Embedded Symmetric Multi Processor Preliminary Data Sheet (Rev. PrB)", [online] http:www.analog.com, 2004.

[85] T. Roll, R. M. Schreier, A. Rothermel, "*DSP-optimierte Bewegungsschätzung für MPEG Videocodierung mittels integraler Projektionen*", ITG/FKTG-Fachtagung "Elektronische Medien", Dortmund, Germany, March 2007; in: VDE-Verlag, ITG-Fachbericht 199, pp. 209-212.

[86] R. M. Schreier, A. Rothermel, "*Verfahren und Vorrichtung zur optimierten prädiktiven Videocodierung",* provisional patent application, DE102005029127A1, DPMA, München, Germany, filing date 23.06.2005, publication date 19.04.2007.

[87] Ralf Michael Schreier, Albrecht Rothermel, "*Method and arrangements for processing video signals*", United States Patent Application 20060291558, USA, filing date 23.06.2006, publication date 28.12.2006.

[88] R. M. Schreier, A. Rothermel, "*Motion Adaptive Intra Refresh for Low-Delay Video Coding*", Digest of Technical Papers IEEE International Conference on Consumer Electronics, ICCE 2006, Las Vegas, USA, 7-11 Jan. 2006, pp. 453-454.

[89] R. M. Schreier, A. Rothermel, "*Motion adaptive intra refresh for the H.264 video coding standard*", IEEE Transactions on Consumer Electronics, Vol. 52, No. 1, Feb. 2006, pp. 249-253.

[90] R. M. Schreier, A. Rothermel, "*Latenzzeitoptimierte Videokompression*", FKTG Jahrestagung, Potsdam, Germany, 15.-18. May 2006.

[91] R. M. Schreier, A. Rothermel, "*A Latency Analysis on H.264 Video Transmission Systems*", invited, IEEE International Conference on Consumer Electronics, ICCE 2008, Las Vegas, USA, 9-13 Jan. 2008; in: Digest of technical papers ICCE 2008, IEEE, pp. 541-521.

[92] R. M. Schreier, A. Rothermel, "*H.264 Echtzeitsysteme mit niedriger Latenzzeit*", FKTG Jahrestagung. München, Germany, 26.-29. May 2008.

[93] R. M. Schreier, A. M. Tushar Iqbal Rahman, G. Krishnamurthy, A. Rothermel, "*Architecture analysis for low-delay video coding*", IEEE International Conference on Multimedia and Expo (ICME), Toronto, Canada, 9-12 July, 2006, in: ISBN 1-4244-0367-7, pp. 2053-2056.

9 LIST OF ABBREVIATIONS

AC	alternating current
AIR	adaptive intra refresh
ARQ	automatic retransmission request
ASP	advanced simple profile
ATM	asynchronous transfer mode
AVC	advanced video coding
AVS	audio and video coding standard
B-frame	bi-directionally predicted frame
BP	baseline profile
bpp	bits per pixel
CBR	Constant bit rate
CD	compact-disc
CDF	cumulated density function
CIE	Commission Internationale de l'Éclairage
CIF	common image format
CMOS	complementary metal-oxide-semiconductor
codec	encoding/decoding device
CPU	central processing unit
CR	conditional replenishment
CRT	cathode ray tube
dB	decibel
DC	direct current
DCT	discrete cosine transformation
DICOM	digital imaging and communications in medicine
DMA	direct memory access
DMB	digital media broadcast
DPCM	differential pulscode modulation
DSP	digital signal processor
DVCAM	digital video camcorder
DVB	digital video broadcast
DVD	digital versatile disc
FIR	finite impulse response

FIU	forced intra update
FMO	flexible macroblock ordering
GOP	group of pictures
HDRC	high dynamic range CMOS
HRD	hypothetical reference decoder
HVS	human visual system
I-frame	intracoded frame
IBBP	video sequence with *IBBPBBP*... picture structure (Table 2.2)
IBP	video sequence with *IBPBPBP*... picture structure (Table 2.2)
IDCT	inverse discrete cosine transformation
IIR	infinite impulse response
IP	video stream encoded only with I and P frames (Table 2.2)
IP2, IP3	video stream using multiple past reference frames (Table 2.2)
IPB	video stream encoded with I, P and B-frames (Table 2.2)
IR	intra refresh
ISO	international standards organization
ITU	international telecommunication union
JM	joint model
JND	just noticeable difference
JPEG	joint photographic experts group
LCD	liquid crystal display
l-opt.	latency-optimized
MBAFF	macroblock-adaptive field-frame coding
MC	motion compensation
ME	motion estimation
ML	main level
MMSE	minimum mean square error
MOST	media oriented systems transport
MP	main profile
MPEG	motion pictures experts group
MP3	MPEG-1, layer 3 audio codec
MSE	mean square error
NTSC	national television systems committee
P-frame	predicted frame

PAL	phase alternating line (video standard)
PCRL	position – component – resolution level (JPEG200 coding mode)
PDF	probability density function
PMV	predicted motion vector
PPI	parallel peripheral interface
PSNR	peak signal to noise ratio
QCIF	quarter common image format
QP	quantization parameter
RD	rate-distortion
RF	radio frequency
RGB	red blue green
RLE	run-length encoding
ROM	read only memory
SAD	sum of absolute differences
SDTV	standard-definition television
SP	simple profile
SR	selected recovery
SSD	sum of squared differences
SSIM	structural similarity measure
TM5	MPEG-2 test model 5 reference implementation
TSS	three step search
TV	television
VBR	variable bit rate
VBV	virtual buffer verifier
VLC	variable length code
WLAN	wireless local area network
WM	windows media
Y	luminance component of video signal
YCbCr	color space defined by luminance Y and color components Cb and Cr
YUV	color space defined by luminance Y and color components U and V

10 LIST OF SYMBOLS

B	buffer size in bits	Eq. 4.12
b_ratio	average frame size ratio for B-frames	Eq. 3.7
C	IR pattern cost	Eq. 5.2
D_{buff}	buffer latency [T_{frame}]	Eq. 4.4
D_{cap}	block capture latency [T_{frame}]	Eq. 4.2
D^{d}_{buff}	initial decoder buffer latency [T_{frame}]	Eq. 4.11
D^{d}_{proc}	decoder processing latency [T_{frame}]	Ch. 4.2
D^{e}_{buff}	initial encoder buffer latency [T_{frame}]	Eq. 4.10
D^{e}_{proc}	encoder processing latency [T_{frame}]	Ch. 4.2
D_{net}	network latency [T_{frame}]	Ch. 4.2
D_{out}	block output latency [T_{frame}]	Ch. 4.2
$D_{reorder}$	reordering latency [T_{frame}]	Eq. 4.7
D_{sys}	system latency [T_{frame}]	Eq. 4.1
D_{tr}	transmission delay	Eq. 4.14
F^{d}	initial decoder buffer occupancy	Eq. 4.11
F^{e}	initial encoder buffer occupancy	Eq. 4.10
K	number of predefined refresh patterns	Ch. 5.2
K_{opt}	index of optimal refresh pattern	Ch. 5.3
N_B	number of B-frames per GOP	Eq. 2.2
N_{block}	processing block size in scan lines	Eq. 4.2
N_R	number of frames in a refresh cycle	Ch. 5.2
N_{GOP}	number of frames per GOP = distance of I-frames	Eq. 2.2
N_P	number of P-frames per GOP	Eq. 2.2
p_ratio	average frame size ratio for P-frames	Eq. 3.6
$PSNR$	peak signal to noise ratio	Eq. 3.5
R	absolute data rate in Mbit/s	Eq. 3.2
r	normalized bit rate in bits per pixel (bpp)	Eq. 3.1
T_{CBR}	CBR time interval	Ch. 4.3
T_{frame}	frame period	Eq. 2.2
T_{GOP}	GOP time interval	Eq. 2.2
t_{line}	scan-line period	Eq. 4.2
$t_{recover}$	error recovery time	Ch. 5.9
T_{slice}	slice time interval	Eq. 4.3

11 LIST OF FIGURES

Figure 2.1	Historical development of video standards.	11
Figure 2.2	Lossy image compression techniques.	14
Figure 2.3	Hybrid video codec transmission system.	16
Figure 2.4	Temporal prediction with IP and IPB coding schemes.	18
Figure 2.5	Effectiveness of basic coding tools.	21
Figure 2.6	Hierarchy of the MPEG-2 data stream.	26
Figure 2.7	Frame and field picture definitions.	30
Figure 3.1	H.264 sequence PSNR at 35 dB operating point.	38
Figure 3.2	MPEG-2 sequence PSNR at 35 dB operating point.	38
Figure 3.3	H.264 average bit rate per frame simulation results (PSNR ≈35 dB).	39
Figure 3.4	H.264 intra coding mode sequence data rates.	41
Figure 3.5	MPEG-2 intra coding mode sequence data rates.	41
Figure 3.6	H.264 intra and IP mode bit rate measures (PSNR ≈ 35 dB, NGOP = 12).	43
Figure 3.7	H.264 p_ratio calculated by the average size ratio of predicted frames and the corresponding intra-coded frames.	43
Figure 3.8	*p_ratio* for H.264 and MPEG-2 (PSNR ≈ 35 dB).	45
Figure 3.9	H.264 forward temporal prediction with multiple past reference frames (PSNR ≈35 dB).	46
Figure 3.10	H.264 bi-directional prediction frame size ratios.	48
Figure 3.11	MPEG-2 bi-directional prediction mode frame size ratios. (NGOP = 12, PSNR ≈ 35 dB).	49
Figure 3.12	H.264 and MPEG-2 rate distortion curves.	50
Figure 3.13	H.264 coding modes quality analysis with respect to IP-coding using only one reference frame.	50
Figure 3.14	Macroblock data rate PDFs and CDFs for selected video sequences (PSNR ≈ 35 dB).	52
Figure 3.15	H.264 slice level rate variations with row-wise slices and randomized slices (PSNR ≈ 35 dB).	55
Figure 4.1	Uncompressed and compressed video transmission systems.	57
Figure 4.2	Detailed video transmission latency model.	58
Figure 4.3	Process-scheduling for intra-frame compressed video transmission.	59
Figure 4.4	Generalized encoder model.	60
Figure 4.5	Generalized decoder model.	63
Figure 4.6	Encoder buffer occupancy and transmission scheduling for IP-coding.	64
Figure 4.7	Process-scheduling for intra-coded video transmission with slice processing.	68

Figure 4.8 Process-scheduling for randomized intra-coded video transmission. 69
Figure 4.9 Process-scheduling for IP-coding. 70
Figure 4.10 Frame reordering for IBBP coding scheme. 71
Figure 4.11 Processing and transmission scheduling for IBBP coding scheme. 72
Figure 4.12 Task scheduling for intracoded frame picture transmission with interlaced video source signal. 73
Figure 4.13 Leaky bucket model for IP-coding with GOP-CBR. 77
Figure 4.14 Derivation of buffer delays from the leaky bucket encoding schedule. 80
Figure 4.15 Intra coding mode buffer analysis (slice resolution). 82
Figure 4.16 Intra coding mode delay analysis. 83
Figure 4.17 IP-mode buffer analysis (slice resolution). 85
Figure 4.18 IP coding mode buffer delay analysis. 86
Figure 4.19 IPB coding mode buffer delay analysis. 88
Figure 4.20 Standard coding mode buffer delay summary (NGOP = 12). 89
Figure 5.1 Forced intra refresh strategies. 92
Figure 5.2 Principle of a region-based intra refresh. 96
Figure 5.3 Intra-refresh border protection. 97
Figure 5.4 Fine-granular and refresh-group intra refresh. 98
Figure 5.5 Twelve-region intra refresh macroblock grouping for CIF resolution. 99
Figure 5.6 The K = 30 refresh patterns used in the simulations. 100
Figure 5.7 Example cost estimation for refresh sequence (1,2,3,4). 102
Figure 5.8 Simulation set up. 104
Figure 5.10 Simulation results for the proposed intra refresh. 106
Figure 5.11 Frame-level program flow of a continuous intra refresh. 110
Figure 5.12 Structural overview of a hybrid video encoder with intra refresh controller. 112
Figure 5.13 Best pattern selection algorithm. 113
Figure 5.14 Region cost estimation. 114
Figure 5.15 Worst-case error recovery for IP and IR coding modes. 116
Figure 5.16 Frame-CBR Intra refresh buffer analysis (slice resolution). 126
Figure 5.17 IR frame CBR delay analysis (NGOP = 12 frames, slice resolution) 127
Figure 5.18 GOP-CBR intra refresh buffer analysis (frame resolution). 130
Figure 5.19 GOP-CBR intra refresh buffer analysis (slice resolution). 131
Figure 5.20 Intra refresh delay analysis with standard and latency-optimized (l-opt.) pattern selection. 132
Figure 5.21 Refresh pattern excess data rate vs. buffer delay estimation. 136
Figure 5.22 Coding mode buffer delay summary (NGOP = 12 frames). 137

Figure 5.23 System delay vs. data rate and recovery time summary. 140
Figure 6.1 Blackfin multimedia DSP Architecture [82]. 144
Figure 6.2 Detailed structure of an MPEG-2 video encoder. 145
Figure 6.3 Program flow options for video encoders. 147
Figure 6.4 IP-mode video encoder frame memory organization. 148
Figure 6.5 Memory organization for windowed motion estimation. 150
Figure 6.6 Wrap-around search window for motion estimation with reduced memory copy operations. 151
Figure C.1 Intra refresh region motion calculation function. 176
Figure C.2 Sequence-level continuous video intra refresh. 177
Figure D.1 IR excess rate measure results for 30 dB simulations. 178
Figure D.2 IR excess rate measure results for 40 dB simulations. 179

12 LIST OF TABLES

Table 2.1	Relationship of lossy video compression techniques and the properties of the human visual system (HVS).	7
Table 2.2	Coding mode name conventions.	20
Table 2.3	Motion estimation tools.	23
Table 2.4	Standard video frame sizes and frame rates.	29
Table 3.1	Characteristics of test sequences.	32
Table 3.2	Quantization parameter adjustment for 35 dB operating point.	37
Table 3.3	Coding gains of H.264 multiple reference frame forward temporal prediction (PSNR ≈ 35 dB).	46
Table 3.4	Characteristic of "large" macroblocks exceeding multiples of the average data rate.	53
Table 4.1	Coding mode summary (analytical results).	75
Table 4.2	Coding mode summary (experimental results).	90
Table 5.1	Optimal refresh pattern selections.	107
Table 5.2	Frame-CBR intra refresh overhead data rates.	125
Table 5.3	Coding mode latency summary.	139
Table 5.4	Coding mode error recovery, bitrate and buffer delays.	141
Table 6.1	Search window reload and move operations.	151
Table A.1	H.264 and MPEG-2 quantizer adjustment for PSNR ≈ 30 dB.	171
Table A.2	H.264 quantizer adjustment for PSNR ≈ 35 dB.	171
Table A.3	MPEG-2 quantizer adjustment for PSNR ≈ 35 dB.	172
Table A.4	H.264 and MPEG-2 quantizer adjustment for PSNR ≈ 40 dB.	172
Table B.1	Intra coding mode delays (slice resolution).	173
Table B.2	Intra random coding mode delays (slice resolution).	173
Table B.3	IP coding mode delays (slice resolution).	173
Table B.4	IP coding mode delays (frame resolution).	174
Table B.5	IBP coding mode delays (frame resolution).	174
Table B.6	IBBP coding mode delays (frame resolution).	174
Table B.7	IR frame CBR delays (slice resolution).	174
Table B.8	IR GOP-CBR delays (slice resolution).	175
Table B.9	IR GOP-CBR delays (frame resolution).	175
Table B.10	Intra refresh delay latency-optimized (slice resolution).	175
Table E.1	Macroblock to region mapping table initialization for $N_R = 12$.	181
Table E.2	Refresh pattern initialization for $N_R = 12$.	181
Table E.3	Macroblock to region mapping table initialization for $N_R = 3$.	182

Table E.4	Refresh pattern initialization for $N_R = 3$.	182
Table E.5	Macroblock to region mapping table initialization for $N_R = 6$.	183
Table E.6	Refresh pattern initialization for $N_R = 6$.	183
Table E.7	Macroblock to region mapping table initialization for $N_R = 24$.	185

Appendix A – H.264 AND MPEG-2 QUALITY ADJUSTMENT

Table A.1: H.264 and MPEG-2 quantizer adjustment for PSNR ≈ 30 dB operating point.

	H.264				MPEG-2			
	QP_{intra}	$QP_{P\text{-}frames}$	$PSNR_{intra}$	$PSNR_{IP}$	QP_{intra}	$QP_{P\text{-}frames}$	$PSNR_{intra}$	$PSNR_{IP}$
canoe	38	36	29.06	29.03	37	35	29.73	29.72
bus	38	37	28.21	28.09	37	38	28.28	28.31
F1 car	38	36	31.29	31.37	37	39	30.83	30.85
foreman	38	37	31.14	31.22	37	38	31.49	31.49
flowergarden	38	36	27.56	27.80	37	43	26.89	26.91
mobile	38	37	26.87	26.71	37	41	26.08	26.09
barcelona	38	36	27.16	27.07	37	38	26.99	27.00
tempete	38	36	27.87	28.04	37	38	27.75	27.78
paris	38	36	28.86	28.91	37	41	28.04	28.06
waterfall	38	37	28.80	28.77	37	37	29.64	29.66
avg. PSNR			**29.81**	**29.87**			**29.86**	**29.87**

Table A.2: H.264 quantizer adjustment for PSNR ≈ 35 dB operating point.

	QP_{intra}	$QP_{P\text{-}frames}$	$QP_{B\text{-}frames}$	$PSNR_{intra}$	$PSNR_{IP}$	$PSNR_{IBP}$	$PSNR_{IBBP}$
canoe	31	29	29	34.18	34.23	34.30	34.26
bus	31	30	30	33.31	33.00	33.10	33.01
F1 car	31	29	29	36.61	36.55	36.67	36.58
foreman	31	30	30	35.55	35.37	35.54	35.51
flowergarden	31	29	29	33.62	33.96	33.92	33.84
mobile	31	29	29	32.71	32.98	33.00	32.90
barcelona	31	29	29	32.32	32.31	32.31	32.18
tempete	31	29	29	33.29	33.43	33.53	33.44
paris	31	29	29	34.20	34.15	34.33	34.32
waterfall	31	30	30	33.18	33.15	33.57	33.56
avg. PSNR				**34.84**	**34.79**	**34.91**	**34.85**

Table A.3: MPEG-2 quantizer adjustment for PSNR ≈ 35 dB operating point.

	QP_{intra}	$QP_{P\text{-}frames}$	$QP_{B\text{-}frames}$	$PSNR_{intra}$	$PSNR_{IP}$	$PSNR_{IBP}$	$PSNR_{IBBP}$
canoe	14	15	14	35.13	35.05	35.17	34.95
bus	14	16	16	33.72	33.62	33.76	33.77
F1 car	14	16	16	36.21	36.15	36.22	36.21
foreman	14	15	15	35.84	35.91	35.88	35.82
flowergarden	14	18	18	33.09	32.96	33.03	32.93
mobile	14	17	17	32.16	32.14	32.14	32.04
barcelona	14	17	17	31.78	31.67	31.77	31.73
tempete	14	16	16	33.16	33.13	33.11	33.02
paris	14	17	18	33.55	33.66	33.53	33.57
waterfall	14	15	16	34.36	34.49	34.41	34.49
avg. PSNR				**34.81**	**34.80**	**34.83**	**34.78**

Table A.4: H.264 and MPEG-2 quantizer adjustment for PSNR ≈ 40 dB operating point.

	H.264				**MPEG-2**			
	QP_{intra}	$QP_{P\text{-}frames}$	$PSNR_{intra}$	$PSNR_{IP}$	QP_{intra}	$QP_{P\text{-}frames}$	$PSNR_{intra}$	$PSNR_{IP}$
canoe	24	23	39.29	38.94	6	7	40.15	40.35
bus	24	23	38.78	38.49	6	8	39.13	38.63
F1 car	24	22	41.33	41.47	6	7	41.02	41.42
foreman	24	23	39.96	39.67	6	7	40.15	40.35
flowergarden	24	23	39.81	39.38	6	9	38.92	38.70
mobile	24	22	38.69	39.12	6	8	37.96	38.19
barcelona	24	22	38.09	38.52	6	8	37.11	37.39
tempete	24	23	38.94	38.55	6	8	38.57	38.34
paris	24	22	39.57	39.58	6	8	38.82	38.94
waterfall	24	23	38.56	38.42	6	7	39.49	39.63
avg. PSNR			**39.78**	**39.77**			**39.67**	**39.89**

Appendix B – SIMULATION RESULTS

Table B.1: Intra coding mode delays (slice resolution).

Buffer delay in T_{frame}	canoe	bus	f1car	foreman	flower-garden	mobile & calendar	barcelona	tempete	paris	waterfall	**max**	**mean**	**median**
D^X_{buff}	0.006	0.050	0.120	0.091	0.000	0.067	0.076	0.002	0.038	0.000	**0.120**	**0.045**	**0.044**
Encoder D^e_{buff}	0.125	0.102	0.017	0.030	0.379	0.000	0.009	0.101	0.035	0.030	**0.379**	**0.083**	**0.033**
Decoder D^d_{buff}	0.130	0.152	0.137	0.121	0.379	0.067	0.085	0.103	0.073	0.030	**0.379**	**0.128**	**0.112**

Table B.2: Intra random coding mode delays (slice resolution).

Buffer delay in T_{frame}	canoe	bus	f1car	foreman	flower-garden	mobile & calendar	barcelona	tempete	paris	waterfall	**max**	**mean**	**median**
	0.003	0.000	0.009	0.010	0.010	0.017	0.006	0.017	0.011	0.020	**0.020**	**0.010**	**0.010**
Encoder D^e_{buff}	0.008	0.021	0.029	0.025	0.008	0.000	0.003	0.003	0.022	0.003	**0.029**	**0.012**	**0.008**
Decoder D^d_{buff}	0.011	0.021	0.038	0.035	0.018	0.017	0.009	0.020	0.033	0.023	**0.038**	**0.022**	**0.020**

Table B.3: IP coding mode delays (slice resolution).

Buffer delay in T_{frame}	canoe	bus	f1car	foreman	flower-garden	mobile & calendar	barcelona	tempete	paris	waterfall	**max**	**mean**	**median**
	0.988	2.275	2.977	3.634	2.017	2.131	2.368	2.065	4.290	6.928	**6.928**	**2.967**	**2.321**
Encoder D^e_{buff}	0.250	0.000	0.000	0.036	0.260	0.000	0.000	0.112	0.000	0.019	**0.260**	**0.068**	**0.010**
Decoder D^d_{buff}	1.237	2.275	2.977	3.670	2.277	2.131	2.368	2.177	4.290	6.948	**6.948**	**3.035**	**2.323**

Appendix B – Simulation Results

Table B.4: IP coding mode delays (frame resolution).

Buffer delay in T_{frame}	canoe	bus	f1car	foreman	flower-garden	mobile & calendar	barcelona	tempete	paris	waterfall	**max**	**mean**	**median**
	0.988	2.275	2.882	3.634	2.017	2.091	2.313	2.065	4.290	6.928	**6.928**	**2.948**	**2.294**
Encoder D^e_{buff}	0.944	0.944	0.944	0.944	0.944	0.944	0.944	0.944	0.944	0.944	**0.944**	**0.944**	**0.944**
Decoder D^d_{buff}	1.932	3.219	3.826	4.579	2.961	3.036	3.258	3.009	5.234	7.873	**7.873**	**3.893**	**3.238**

Table B.5: IBP coding mode delays (frame resolution).

Buffer delay in T_{frame}	canoe	bus	f1car	foreman	flower-garden	mobile & calendar	barcelona	tempete	paris	waterfall	**max**	**mean**	**median**
	0.969	2.606	2.917	4.325	2.235	2.663	3.070	2.882	4.462	6.950	**6.950**	**3.308**	**2.899**
Encoder D^e_{buff}	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	**1.000**	**1.000**	**1.000**
Decoder D^d_{buff}	1.969	3.606	3.917	5.325	3.235	3.663	4.070	3.882	5.462	7.950	**7.950**	**4.308**	**3.899**

Table B.6: IBBP coding mode delays (frame resolution).

Buffer delay in T_{frame}	canoe	bus	f1car	foreman	flower-garden	mobile & calendar	barcelona	tempete	paris	waterfall	**max**	**mean**	**median**
	0.863	2.517	2.850	4.150	2.159	2.766	3.407	2.907	4.375	7.059	**7.059**	**3.305**	**2.878**
Encoder D^e_{buff}	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	**1.000**	**1.000**	**1.000**
Decoder D^d_{buff}	1.863	3.517	3.850	5.150	3.159	3.766	4.407	3.907	5.375	8.059	**8.059**	**4.305**	**3.878**

Table B.7: IR frame CBR delays (slice resolution).

Buffer delay in T_{frame}	canoe	bus	f1car	foreman	flower-garden	mobile & calendar	barcelona	tempete	paris	waterfall	**max**	**mean**	**median**
	0.000	0.177	0.357	0.284	0.000	0.212	0.286	0.012	0.235	0.413	**0.413**	**0.198**	**0.224**
D^e_{buff}	0.337	0.187	0.095	0.307	0.479	0.151	0.035	0.310	0.355	0.498	**0.498**	**0.275**	**0.309**
D^d_{buff}	0.337	0.363	0.452	0.591	0.479	0.363	0.322	0.322	0.591	0.911	**0.911**	**0.473**	**0.407**

Table B.8: IR GOP-CBR delays (slice resolution).

Buffer delay in T_{frame}	canoe	bus	f1car	foreman	flower-garden	mobile & calendar	barcelona	tempete	paris	waterfall	**max**	**mean**	**median**
	0.141	0.345	0.483	0.688	0.794	0.426	0.419	0.192	0.547	1.076	**1.076**	**0.511**	**0.454**
Encoder D^{e}_{buff}	0.461	0.356	0.208	0.771	1.208	0.356	0.163	0.458	0.641	1.169	**1.208**	**0.579**	**0.460**
Decoder D^{d}_{buff}	0.603	0.700	0.691	1.459	2.002	0.781	0.582	0.650	1.188	2.245	**2.245**	**1.090**	**0.741**

Table B.9: IR GOP-CBR delays (frame resolution).

Buffer delay in T_{frame}	canoe	bus	f1car	foreman	flower-garden	mobile & calendar	barcelona	tempete	paris	waterfall	**max**	**mean**	**median**
	0.141	0.182	0.128	0.547	0.794	0.239	0.156	0.181	0.331	0.703	**0.794**	**0.340**	**0.211**
Encoder D^{e}_{buff}	1.086	1.126	1.073	1.519	1.739	1.184	1.099	1.125	1.276	1.647	**1.739**	**1.287**	**1.155**
Decoder D^{d}_{buff}	1.227	1.308	1.201	2.066	2.533	1.423	1.255	1.306	1.607	2.350	**2.533**	**1.628**	**1.365**

Table B.10: Intra refresh delay latency-optimized (slice resolution).

Buffer delay in T_{frame}	canoe	bus	f1car	foreman	flower-garden	mobile & calendar	barcelona	tempete	paris	waterfall	**max**	**mean**	**median**
	0.062	0.278	0.380	0.353	0.311	0.295	0.336	0.121	0.269	0.371	**0.380**	**0.278**	**0.303**
Encoder D^{e}_{buff}	0.301	0.178	0.086	0.250	0.444	0.050	0.000	0.268	0.354	0.466	**0.466**	**0.240**	**0.259**
Decoder D^{d}_{buff}	0.363	0.456	0.467	0.603	0.755	0.346	0.336	0.388	0.623	0.837	**0.837**	**0.517**	**0.461**

Appendix C – Intra Refresh Algorithm Flowcharts

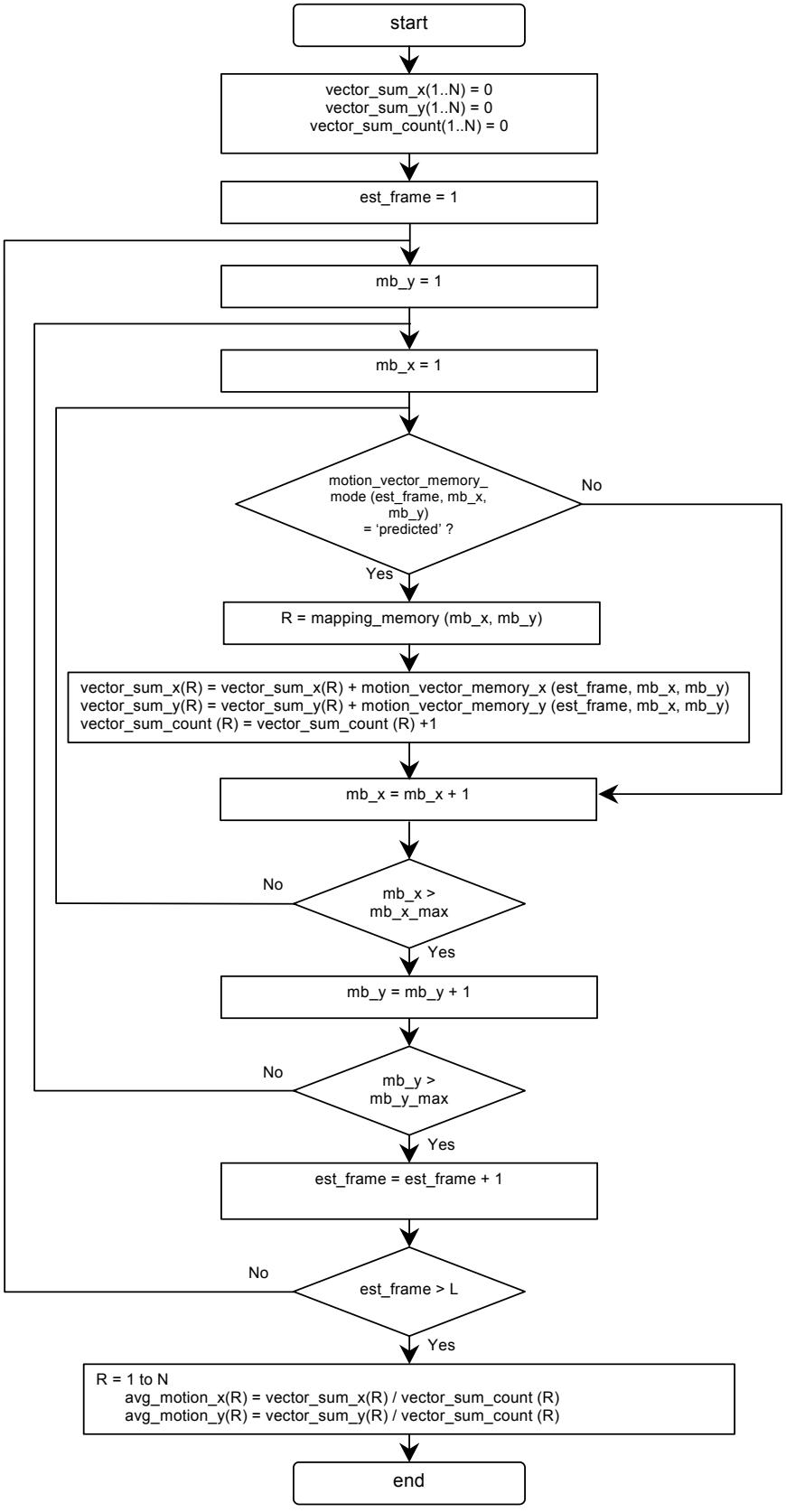


Figure C.1: Intra refresh region motion calculation function.

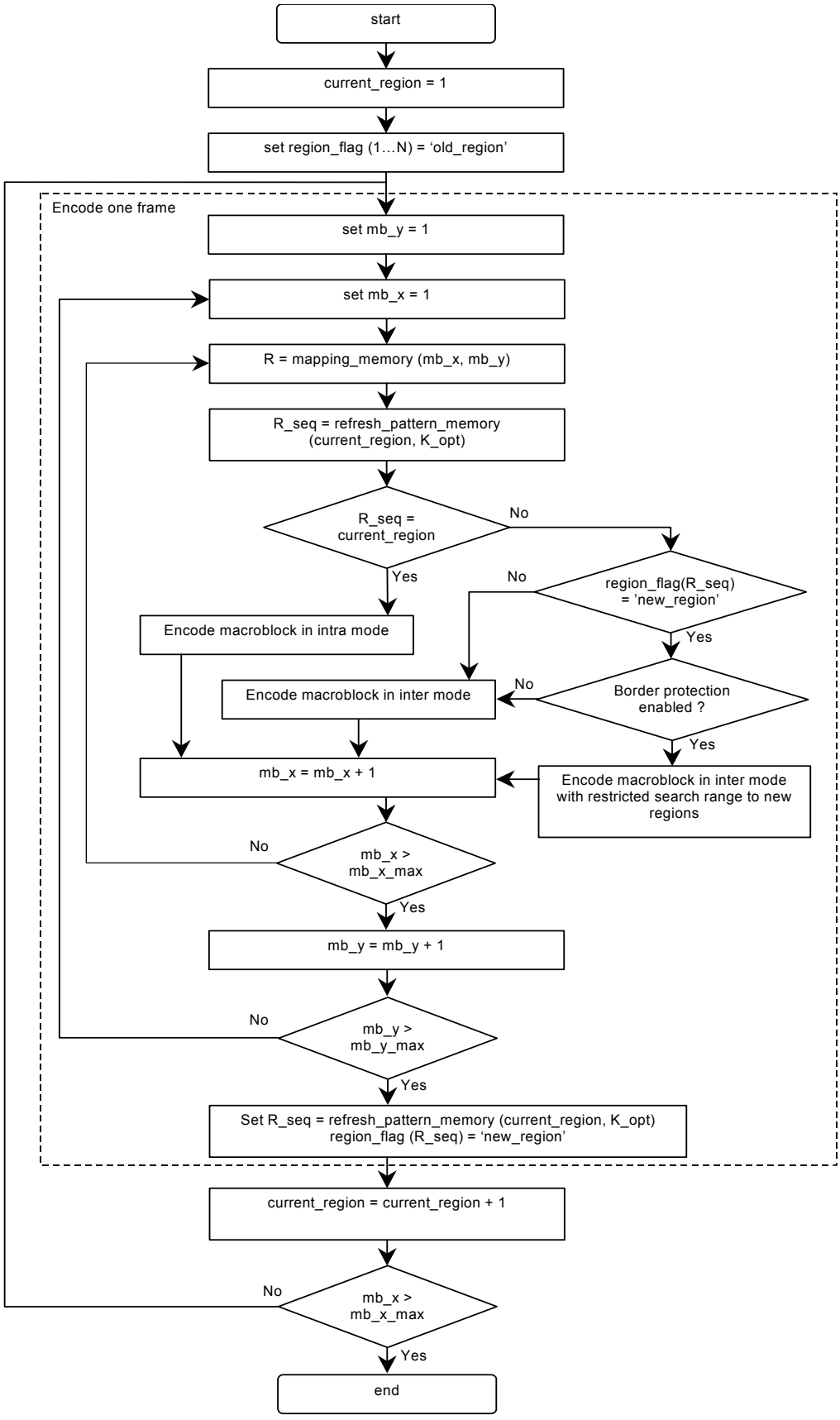


Figure C.2: Sequence-level continuous video intra refresh.

Appendix D – Intra Refresh Excess Data Rate Simulations

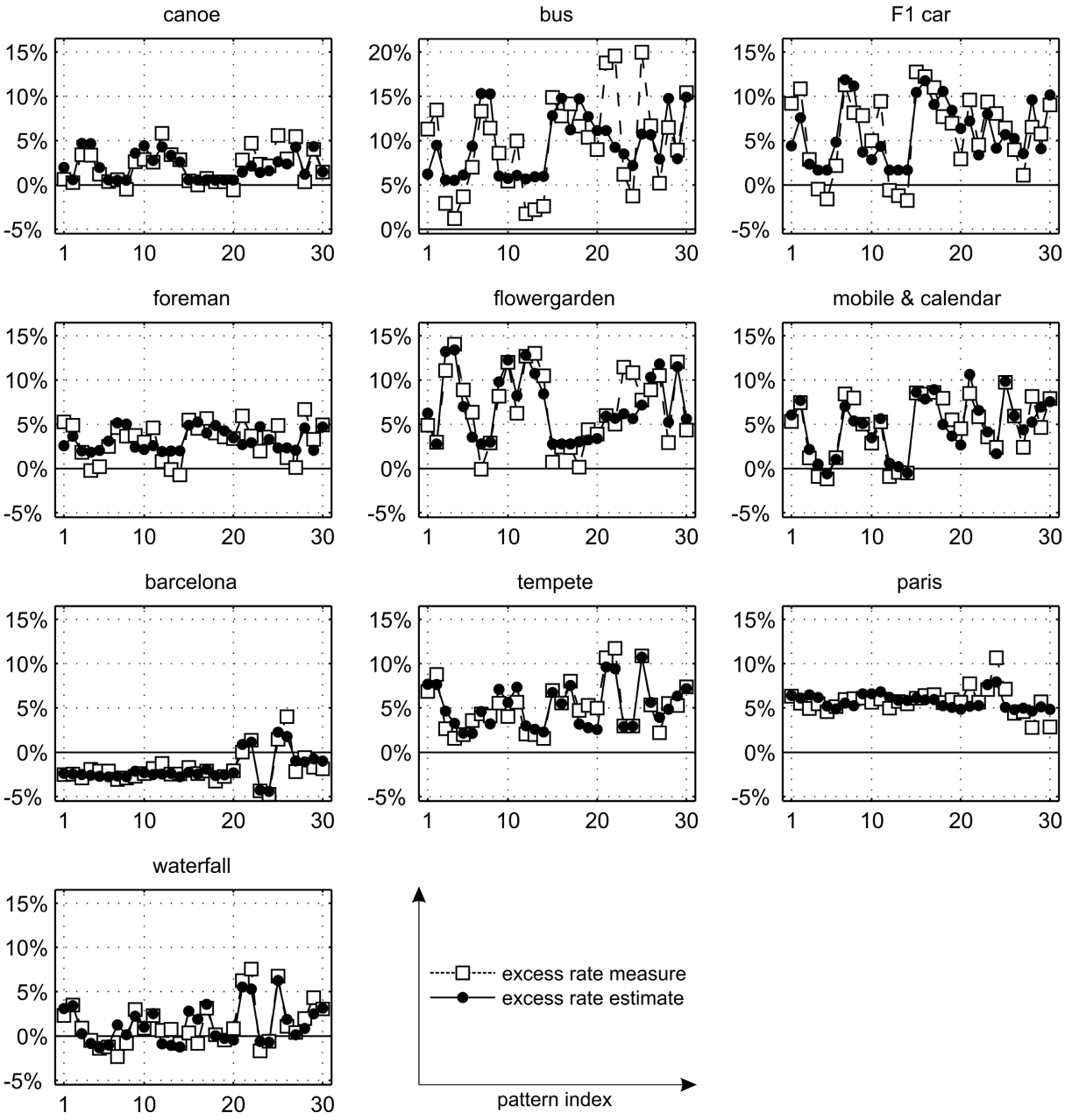


Figure D.1: IR excess rate measure results for 30 dB simulations.

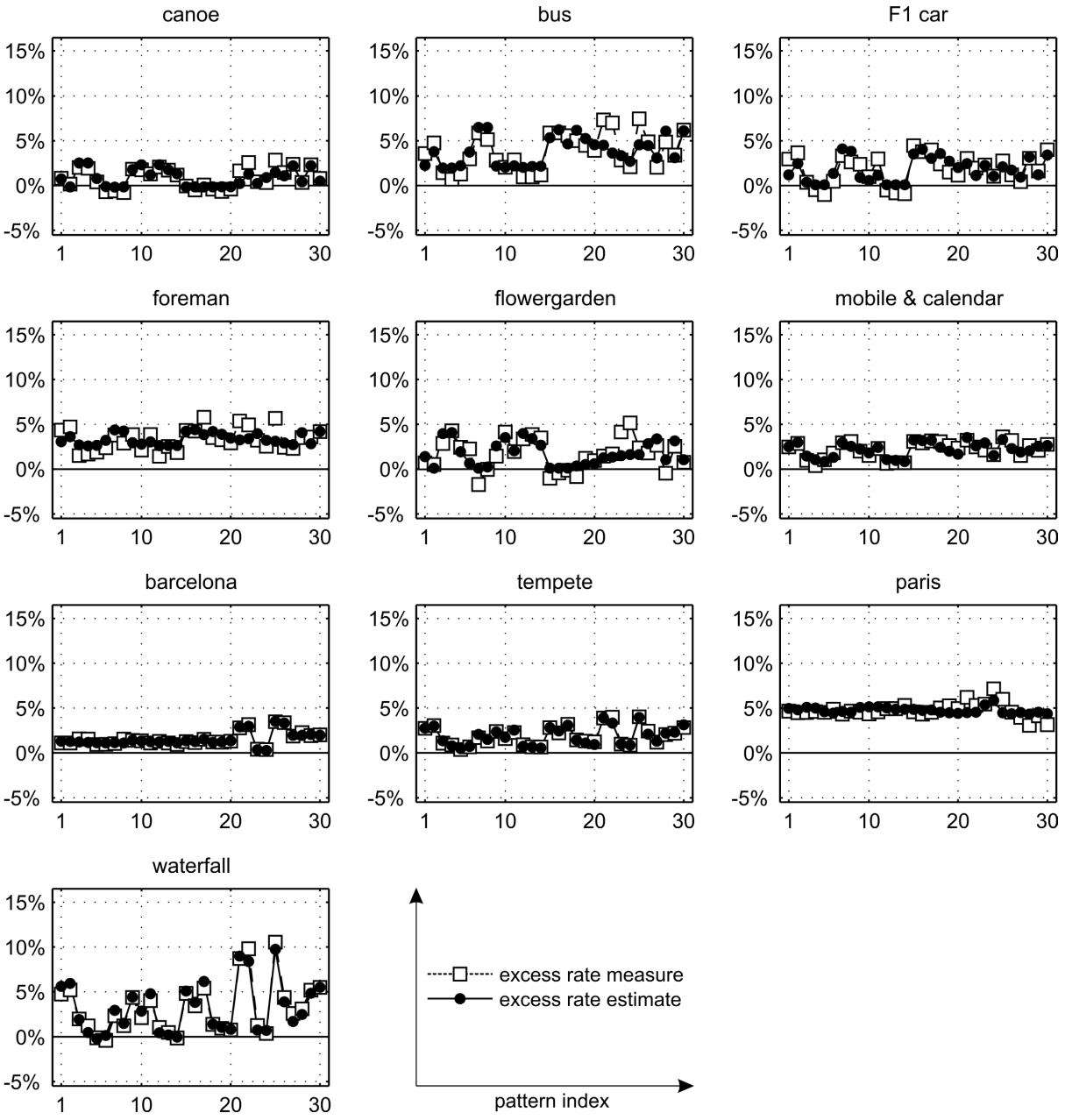


Figure D.2: IR excess rate measure results for 40 dB simulations.

Appendix E – INTRA REFRESH TABLE INITIALIZATIONS

Table E.1: Macroblock to region mapping table initialization for $N_R = 12$.

Macroblock y-coordinate	Macroblock x-coordinate																					
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
1	0	0	0	0	0	0	1	1	1	1	1	2	2	2	2	2	2	3	3	3	3	3
2	0	0	0	0	0	0	1	1	1	1	1	2	2	2	2	2	2	3	3	3	3	3
3	0	0	0	0	0	0	1	1	1	1	1	2	2	2	2	2	2	3	3	3	3	3
4	0	0	0	0	0	0	1	1	1	1	1	2	2	2	2	2	2	3	3	3	3	3
5	0	0	0	0	0	0	1	1	1	1	1	2	2	2	2	2	2	3	3	3	3	3
6	0	0	0	0	0	0	1	1	1	1	1	2	2	2	2	2	2	3	3	3	3	3
7	4	4	4	4	4	4	5	5	5	5	5	6	6	6	6	6	6	7	7	7	7	7
8	4	4	4	4	4	4	5	5	5	5	5	6	6	6	6	6	6	7	7	7	7	7
9	4	4	4	4	4	4	5	5	5	5	5	6	6	6	6	6	6	7	7	7	7	7
10	4	4	4	4	4	4	5	5	5	5	5	6	6	6	6	6	6	7	7	7	7	7
11	4	4	4	4	4	4	5	5	5	5	5	6	6	6	6	6	6	7	7	7	7	7
12	4	4	4	4	4	4	5	5	5	5	5	6	6	6	6	6	6	7	7	7	7	7
13	8	8	8	8	8	8	9	9	9	9	9	10	10	10	10	10	10	11	11	11	11	11
14	8	8	8	8	8	8	9	9	9	9	9	10	10	10	10	10	10	11	11	11	11	11
15	8	8	8	8	8	8	9	9	9	9	9	10	10	10	10	10	10	11	11	11	11	11
16	8	8	8	8	8	8	9	9	9	9	9	10	10	10	10	10	10	11	11	11	11	11
17	8	8	8	8	8	8	9	9	9	9	9	10	10	10	10	10	10	11	11	11	11	11
18	8	8	8	8	8	8	9	9	9	9	9	10	10	10	10	10	10	11	11	11	11	11

Table E.2: Refresh pattern initialization for $N_R = 12$ (as illustrated in figure 5.6).

Pattern	Frame												Note
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	
1	0	1	2	3	4	5	6	7	8	9	10	11	lines top left to bottom right
2	3	2	1	0	7	6	5	4	11	10	9	8	lines top right to bottom left
3	0	4	8	1	5	9	2	6	10	3	7	11	columns top left to bottom right
4	8	4	0	9	5	1	10	6	2	11	7	3	columns bottom left to to right
5	8	9	10	11	4	5	6	7	0	1	2	3	lines bottom left to top right
6	11	10	9	8	7	6	5	4	3	2	1	0	lines bottom right to top left
7	3	7	11	2	6	10	1	5	9	0	4	8	columns top right to bottom left
8	11	7	3	10	6	2	9	5	1	8	4	0	columns bottom right to top left
9	0	1	4	2	5	8	3	6	9	7	10	11	diagonal top left
10	0	4	1	8	5	2	9	6	3	10	7	11	
11	0	1	2	4	3	5	8	6	7	9	10	11	
12	8	4	9	0	5	10	1	6	11	2	7	3	diagonal bottom left
13	8	9	4	10	5	0	11	6	1	7	2	3	
14	8	9	10	4	11	5	0	6	7	1	2	3	
15	3	2	7	1	6	11	0	5	10	4	9	8	diagonal top right
16	3	7	2	11	6	1	10	5	0	9	4	8	
17	3	2	1	7	6	0	5	11	4	10	9	8	
18	11	7	10	3	6	9	2	5	8	1	4	0	diagonal bottom right
19	11	10	7	9	6	3	8	5	2	4	1	0	
20	11	10	9	7	6	8	5	3	4	2	1	0	
21	5	6	7	3	2	1	0	4	8	9	10	11	box-out counter-clockwise
22	6	5	4	0	1	2	3	7	11	10	9	8	box-out clockwise
23	8	9	10	11	7	3	2	1	0	4	5	6	box-in counter-clockwise
24	11	10	9	8	4	0	1	2	3	7	6	5	box-in clockwise
25	5	6	1	2	4	0	7	3	9	10	11	8	
26	5	6	9	10	4	8	7	11	1	2	3	0	
27	5	9	8	4	1	0	6	10	11	7	2	3	
28	6	10	11	7	2	3	5	9	8	4	1	0	
29	5	1	0	4	9	8	6	2	3	7	10	11	
30	6	2	3	7	10	11	5	1	0	4	8	9	

It should be noted that there are two ways to organize the refresh pattern table. The described version will associate each frame in a refresh sequence with a corresponding refresh region which is refreshed in this frame. It would also be possible to associate each refresh region with a corresponding frame in which it is refreshed. The first implementation has the advantage that dirty regions can be easily identified by comparing the current frame number with the refresh pattern table number of the region in which a macroblock is located. This eases the detection of dirty regions especially if the algorithm is used with multiple previous reference frames.

Appendix E – Intra Refresh Table Initializations

Table E.3: Macroblock to region mapping table initialization for $N_R = 3$.

		Macroblock x-coordinate																				
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
Macroblock y-coordinate	1	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	2	2	2	2	2	2	2
	2	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	2	2	2	2	2	2	2
	3	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	2	2	2	2	2	2	2
	4	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	2	2	2	2	2	2	2
	5	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	2	2	2	2	2	2	2
	6	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	2	2	2	2	2	2	2
	7	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	2	2	2	2	2	2	2
	8	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	2	2	2	2	2	2	2
	9	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	2	2	2	2	2	2	2
	10	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	2	2	2	2	2	2	2
	11	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	2	2	2	2	2	2	2
	12	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	2	2	2	2	2	2	2
	13	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	2	2	2	2	2	2	2
	14	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	2	2	2	2	2	2	2
	15	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	2	2	2	2	2	2	2
	16	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	2	2	2	2	2	2	2
	17	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	2	2	2	2	2	2	2
	18	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	2	2	2	2	2	2	2

Table E.4: Refresh pattern initialization for $N_R = 3$.

Pattern	**Frame**		
	1	**2**	**3**
1	0	1	2
2	0	2	1
3	2	1	0
4	2	0	1
5	1	0	2
6	1	2	0

Table E.5: Macroblock to region mapping table initialization for $N_R = 6$.

		Macroblock x-coordinate																					
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
Macroblock y-coordinate	1	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	2	2	2	2	2	2	2
	2	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	2	2	2	2	2	2	2
	3	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	2	2	2	2	2	2	2
	4	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	2	2	2	2	2	2	2
	5	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	2	2	2	2	2	2	2
	6	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	2	2	2	2	2	2	2
	7	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	2	2	2	2	2	2	2
	8	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	2	2	2	2	2	2	2
	9	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	2	2	2	2	2	2	2
	10	3	3	3	3	3	3	3	4	4	4	4	4	4	4	4	5	5	5	5	5	5	5
	11	3	3	3	3	3	3	3	4	4	4	4	4	4	4	4	5	5	5	5	5	5	5
	12	3	3	3	3	3	3	3	4	4	4	4	4	4	4	4	5	5	5	5	5	5	5
	13	3	3	3	3	3	3	3	4	4	4	4	4	4	4	4	5	5	5	5	5	5	5
	14	3	3	3	3	3	3	3	4	4	4	4	4	4	4	4	5	5	5	5	5	5	5
	15	3	3	3	3	3	3	3	4	4	4	4	4	4	4	4	5	5	5	5	5	5	5
	16	3	3	3	3	3	3	3	4	4	4	4	4	4	4	4	5	5	5	5	5	5	5
	17	3	3	3	3	3	3	3	4	4	4	4	4	4	4	4	5	5	5	5	5	5	5
	18	3	3	3	3	3	3	3	4	4	4	4	4	4	4	4	5	5	5	5	5	5	5

Table E.6: Refresh pattern initialization for $N_R = 6$.

Pattern	**Frame**					
	1	**2**	**3**	**4**	**5**	**6**
1	0	1	2	3	4	5
2	2	1	0	5	4	3
3	3	4	5	0	1	2
4	5	4	3	2	1	0
5	0	3	1	4	2	5
6	3	0	4	1	5	2
7	2	5	1	4	0	3
8	5	2	4	1	3	0
9	0	1	3	2	4	5
10	3	4	0	5	1	2
11	2	1	5	0	4	3
12	5	4	2	3	1	0
13	0	3	2	5	1	4
14	2	5	0	3	1	4
15	3	0	5	2	4	1
16	5	2	3	0	4	1
17	1	4	0	3	2	5
18	4	1	3	0	5	2
19	1	4	3	0	2	5
20	1	4	5	2	0	3
21	4	1	0	3	5	2
22	4	1	2	5	3	0

Table E.7: Macroblock to region mapping table initialization for $N_R = 24$.

		Macroblock x-coordinate																					
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
Macroblock y-coordinate	1	0	0	0	0	0	0	1	1	1	1	1	2	2	2	2	2	2	3	3	3	3	3
	2	0	0	0	0	0	1	1	1	1	1	1	2	2	2	2	2	3	3	3	3	3	3
	3	0	0	0	0	0	0	1	1	1	1	1	2	2	2	2	2	2	3	3	3	3	3
	4	4	4	4	4	4	4	5	5	5	5	5	6	6	6	6	6	6	7	7	7	7	7
	5	4	4	4	4	4	5	5	5	5	5	5	6	6	6	6	6	7	7	7	7	7	7
	6	4	4	4	4	4	4	5	5	5	5	5	6	6	6	6	6	6	7	7	7	7	7
	7	8	8	8	8	8	8	9	9	9	9	9	10	10	10	10	10	10	11	11	11	11	11
	8	8	8	8	8	8	9	9	9	9	9	9	10	10	10	10	10	11	11	11	11	11	11
	9	8	8	8	8	8	8	9	9	9	9	9	10	10	10	10	10	10	11	11	11	11	11
	10	12	12	12	12	12	12	13	13	13	13	13	14	14	14	14	14	14	15	15	15	15	15
	11	12	12	12	12	12	13	13	13	13	13	13	14	14	14	14	14	15	15	15	15	15	15
	12	12	12	12	12	12	12	13	13	13	13	13	14	14	14	14	14	14	15	15	15	15	15
	13	16	16	16	16	16	16	17	17	17	17	17	18	18	18	18	18	18	19	19	19	19	19
	14	16	16	16	16	16	17	17	17	17	17	17	18	18	18	18	18	19	19	19	19	19	19
	15	16	16	16	16	16	16	17	17	17	17	17	18	18	18	18	18	18	19	19	19	19	19
	16	20	20	20	20	20	20	21	21	21	21	21	22	22	22	22	22	22	23	23	23	23	23
	17	20	20	20	20	20	21	21	21	21	21	21	22	22	22	22	22	23	23	23	23	23	23
	18	20	20	20	20	20	20	21	21	21	21	21	22	22	22	22	22	22	23	23	23	23	23

Table E.8: Refresh pattern initialization for $N_R = 24$.

Pat-tern	Frame																							
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**
1	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
	lines top left to bottom right																							
2	5	4	3	2	1	0	11	10	9	8	7	6	17	16	15	14	13	12	23	22	21	20	19	18
	lines top right to bottom left																							
3	0	6	12	18	1	7	13	19	2	8	14	20	3	9	15	21	4	10	16	22	5	11	17	23
	colums top left to bottom right																							
4	18	12	6	0	19	13	7	1	20	14	8	2	21	15	9	3	22	16	10	4	23	17	11	5
	colums bottom left to to right																							
5	18	19	20	21	22	23	12	13	14	15	16	17	6	7	8	9	10	11	0	1	2	3	4	5
	lines bottom left to top right																							
6	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	lines bottom right to top left																							
7	5	11	17	23	4	10	16	11	3	9	15	21	2	8	14	20	1	7	13	19	0	6	12	18
	colums top right to bottom left																							
8	23	17	11	5	22	16	10	4	21	15	9	3	20	14	8	2	19	13	7	1	18	12	6	0
	colums bottom right to top left																							
9	0	1	6	2	7	12	3	8	13	18	4	9	14	19	5	10	15	20	11	16	21	17	22	23
	diagonal top left																							
10	0	6	1	12	7	2	18	13	8	3	19	14	9	4	20	15	10	5	21	16	11	22	17	23
11	0	1	2	6	3	7	4	8	12	5	9	13	10	14	18	11	15	19	16	20	17	21	22	23
12	18	12	19	6	13	20	0	7	14	21	1	8	15	22	2	9	16	23	3	10	17	4	11	5
	diagonal bottom left																							
13	18	19	12	20	13	6	21	14	7	0	22	15	8	1	23	16	9	2	17	10	3	11	4	5
14	18	19	20	12	21	13	22	14	6	23	15	7	16	8	0	17	9	1	10	2	11	3	4	5
15	5	4	11	3	10	17	2	9	16	23	1	8	15	22	0	7	14	21	6	13	20	12	19	18
	diagonal top right																							
16	5	11	4	17	10	3	23	16	9	2	22	15	8	1	21	14	7	0	20	13	6	19	12	18
17	5	4	3	11	2	10	1	9	17	0	8	16	7	15	23	6	14	22	13	21	12	20	19	18
18	23	17	22	11	16	21	5	10	15	20	4	9	14	19	3	8	13	18	2	7	12	1	6	0
	diagonal bottom right																							
19	23	22	17	21	16	11	20	15	10	5	19	14	9	4	18	13	8	3	12	7	2	6	1	0
20	23	22	21	17	20	16	19	15	11	18	14	10	13	9	5	12	8	4	7	3	6	2	1	0
21	9	8	7	13	15	15	16	10	4	3	2	1	0	6	12	18	19	20	21	22	23	17	11	5
	box-out counter-clockwise																							
22	15	14	13	7	8	9	10	16	22	21	20	29	18	12	6	0	1	2	3	4	5	11	17	23
	box-out clockwise																							
23	18	19	20	21	22	23	17	11	5	4	3	2	1	0	6	12	13	14	15	16	10	9	8	7
	box-in counter-clockwise																							
24	23	22	21	20	19	18	12	6	0	1	2	3	4	5	11	17	16	15	14	13	7	8	9	10
	box-in clockwise																							
25	8	7	2	1	6	0	9	10	3	4	11	5	14	13	20	19	12	18	5	16	21	22	17	23
26	9	10	3	4	11	5	8	7	2	1	6	0	15	16	21	22	17	23	14	13	20	19	12	18
27	14	13	20	19	12	18	8	7	2	1	6	0	15	16	21	22	17	23	9	10	3	4	11	5
28	15	16	21	22	17	23	9	10	3	4	11	5	14	13	20	19	12	18	8	7	2	1	6	0
29	14	13	8	7	2	1	20	19	12	6	18	0	15	16	9	10	3	4	21	22	17	11	13	5
30	15	16	9	10	3	4	21	22	17	11	13	5	14	13	8	7	2	1	20	19	12	6	18	0

Appendix F – ENCODER CONFIGURATION

H.264 encoder configuration

JM7.5b encoder configuration file (excerpt of important sections)

```
###############################################
# Files
###############################################
InputFile                = "barcelona_cif.yuv"  # Input sequence, YUV 4:2:0
InputHeaderLength        = 152064 # If the inputfile has a header, state
                                  it's length in byte here
FramesToBeEncoded        = 72    # Number of frames to be coded
PictureRate              = 30    # Picture Rate per second (1-100)
SourceWidth              = 352   # Image width in Pels, must be multiple of
                                  16
SourceHeight             = 288   # Image height in Pels, must be multiple of
                                  16
TraceFile                = ""
ReconFile                = "barcelona/barcelona_ip.yuv"
OutputFile               = "barcelona/barcelona_ip.264"

###############################################
# Encoder Control
###############################################

IntraPeriod              = 12    #0  Period of I-Frames (0=only first)
IDRIntraEnable           =  0    # Force IDR Intra  (0=disable 1=enable)
QPFirstFrame             = 31    #   QP(intra):   Quant.   param   for   intra
                                  slices(0-51)
QPRemainingFrame         = 29    # QP(inter): Quant. param for inter slices
                                  (0-51)
FrameSkip                =  0    #1  Number of frames to be skipped in input
                                  (e.g 2 will code every third frame)
UseHadamard              =  0    #1  Hadamard transform (0=not used, 1=used)
SearchRange              = 16    #16 Max search range
NumberReferenceFrames    =  1    #5  Number of previous frames used for inter
                                  motion search (1-5)
MbLineIntraUpdate        =  0    #0  Error robustness(extra intra macro block
                                  updates)(0=off, N: One GOB every N frames
                                  are intra coded)
RandomIntraMBRefresh     =  0    #0  Forced intra MBs per picture
InterSearch16x16         =  1    #1   Inter  block  search  16x16  (0=disable,
                                  1=enable)
InterSearch16x8          =  1    #1  Inter block search 16x8  (0=disable,
InterSearch8x16          =  1    #1  Inter block search  8x16 (0=disable,
InterSearch8x8           =  1    #1  Inter block search  8x8  (0=disable,
InterSearch8x4           =  1    #1  Inter block search  8x4  (0=disable,
InterSearch4x8           =  1    #1  Inter block search  4x8  (0=disable,
InterSearch4x4           =  1    #1  Inter block search  4x4  (0=disable,

RestrictSearchRange      =  2    #2  restriction for (0: blocks and ref, 1:
                                  ref, 2: no restrictions)
RDOptimization           =  1    #1  rd-optimized mode decision (0:off, 1:on,
                                  2: with losses)
UseConstrainedIntraPred  =  1    #0   If  1,  Inter  pixels  are  not  used  for
                                  Intra macroblock prediction.
LoopFilterParametersFlag = 0     # Configure loop filter (0=parameter below
                                  ingored, 1=parameters sent)
LoopFilterDisable        = 0     #  Disable  loop  filter  in  slice  header
                                  (0=Filter, 1=No Filter)
LoopFilterAlphaC0Offset  = -2    # Alpha & C0 offset div. 2, {-6, -5, ... 0,
                                  +1, .. +6}
LoopFilterBetaOffset     = -1    # Beta offset div. 2, {-6, -5, ... 0, +1, ..
                                  +6}
```

MPEG-2 Encoder Configuration

```
MPEG-2 Test Sequence, 25 frames/sec

barcelona_cif  /* name of source files without ext. */
sim11_mpeg2_barcelona_ip      /* name of reconstructed images  */
-         /* name of intra quant matrix file     ("-": default matrix) */
-         /* name of non intra quant matrix file ("-": default matrix) */
sim11_mpeg2_barcelona_ip.txt  /* name of statistics file ("-": stdout ) */
4         /* input picture file format: 0=*.Y,*.U,*.V, 1=*.yuv, 2=*.ppm
             ,3=.yuv(yuyv stream), 4=.yuv(yuv stream)*/
72        /* number of frames */
1         /* number of first frame */
00:00:00:00 /* timecode of first frame */
12        /* N (# of frames in GOP) */
1         /* M (I/P frame distance) */
0         /* ISO/IEC 11172-2 stream */
0         /* 0:frame pictures, 1:field pictures */
352       /* horizontal_size */
288       /* vertical_size */
2         /* aspect_ratio_information 1=square pel, 2=4:3, 3=16:9, 4=2.11:1
*/
3         /* frame_rate_code 1=23.976, 2=24, 3=25, 4=29.97, 5=30 fps */
2000000.0 /* bit_rate (bits/s) */
112       /* vbv_buffer_size (in multiples of 16 kbit) */
0         /* low_delay  */
0         /* constrained_parameters_flag */
4         /* Profile ID: Simple = 5, Main = 4, SNR = 3, Spatial = 2,
             High = 1 */
8         /* Level ID:   Low = 10, Main = 8, High 1440 = 6, High = 4 */
0         /* progressive_sequence */
1         /* chroma_format: 1=4:2:0, 2=4:2:2, 3=4:4:4 */
1         /* video_format: 0=comp., 1=PAL, 2=NTSC, 3=SECAM, 4=MAC*/
5         /* color_primaries */
5         /* transfer_characteristics */
5         /* matrix_coefficients */
352       /* display_horizontal_size */
288       /* display_vertical_size */
0         /* intra_dc_precision (0: 8 bit, 1: 9 bit, 2: 10 bit, 3: 11 bit*/
1         /* top_field_first */
1 1 1     /* frame_pred_frame_dct (I P B) */
0 0 0     /* concealment_motion_vectors (I P B) */
0 0 0     /* q_scale_type  (I P B) */
14 17 14  /* q_scale_value  (I P B) */
1 0 0     /* intra_vlc_format (I P B)*/
0 0 0     /* alternate_scan (I P B) */
0         /* repeat_first_field */
1         /* progressive_frame */
0         /* P distance between complete intra slice refresh */
0         /* rate control: r (reaction parameter) */
0         /* rate control: avg_act (initial average activity) */
0         /* rate control: Xi (initial I frame global complexity measure)*/
0         /* rate control: Xp (initial P frame global complexity measure)*/
0         /* rate control: Xb (initial B frame global complexity measure)*/
0         /* rate control: d0i (initial I frame virtual buffer fullness) */
0         /* rate control: d0p (initial P frame virtual buffer fullness) */
0         /* rate control: d0b (initial B frame virtual buffer fullness) */
3 3 16 16 /* P:  forw_hor_f_code forw_vert_f_code search_width/height */
3 3 16 16 /* B1: forw_hor_f_code forw_vert_f_code search_width/height */
3 3 16 16 /* B1: back_hor_f_code back_vert_f_code search_width/height */
3 3 16 16 /* B2: forw_hor_f_code forw_vert_f_code search_width/height */
3 3 16 16 /* B2: back_hor_f_code back_vert_f_code search_width/height */
```