DISSERTATION

# Error Resilient Transmission of Video Streaming over Wireless Mobile Networks

ausgeführt zum Zwecke der Erlangung des akademischen Grades
eines Doktors der technischen Wissenschaften

eingereicht an der Technischen Universität Wien
Fakultät für Elektrotechnik und Informationstechnik

von

**DI Olivia Nemethova**
Dr. G. Praderstrasse 13/10
A-2103 Langenzersdorf

geboren am 2. Februar 1977 in Bratislava (SK)
Matrikelnummer: 0326971

Wien, im Mai 2007

# Begutachter:

**Univ. Prof. Dr. Markus Rupp**
Institut für Nachrichtentechnik und Hochfrequenztechnik
Technische Universität Wien
Österreich

**Dr. Michel Kieffer**
Laboratoire des signaux et systèmes
Supélec, École supérieure d'électricité
Université Paris XI Orsay
Frankreich

# Abstract

THE third generation of mobile systems brought higher data rates that allow for provisioning of multimedia services containing also video. The real-time services like video call, conferencing, and streaming are particularly challenging for mobile communication systems due to the wireless channel quality variations. The mechanism for video compression utilizes a hybrid of temporal and spatial prediction, transform coding and variable length coding. The combination of these methods provides high compression gain, but at the same time makes the encoded stream more prone to errors.

In this thesis, techniques for error resilient transmission of video streaming over wireless mobile networks are investigated. Focus is given to the recent H.264/AVC standard, although the majority of the proposed method apply to other video coding standards, too. The first part is dedicated to exploiting the residual redundancy of the received video stream at the decoder. The redundancy is used for error localization within a damaged packet that would be discarded otherwise. Error detection using syntax analysis and detection of impairments in the picture domain do not require any additional rate while reducing the resulting distortion. As their detection performance is limited, alternative methods assisted by an encoder are further proposed. The experimental results show that particularly the resynchronization of the variable length code, facilitated by out-of-stream signalized side information, improves the performance considerably.

In the second part of this thesis, error concealment methods are reviewed, compared, improved and novel are proposed. There is no single error concealment method performing best in all situations. Temporal interpolation typically performs better than spatial, unless scene change occurs. Therefore, an adaptive mechanism is proposed that chooses from several temporal and spatial methods according to the situation. A scene change detector based on dynamic thresholding, working independently of encoder settings, is also a part of the proposal. In spite of its low complexity, the proposed mechanism provides an impressive quality gain.

The final part handles the cross-layer design for UMTS (Universal Mobile Telecommunications System). It is shown that utilizing the radio link information at the application layer of the receiver can substantially improve the quality at practically no costs (no rate increase, no additional complexity). Moreover, it is shown, that UMTS radio link layer errors are predictable. Differentiation between the intervals with high and low probability of error allows for assignment of different priority level to the video packets of one user, based on semantic information or expected distortion. The scheduler disposing with such information considerably improves the resulting video quality.

# Kurzfassung

DURCH die Einführung der dritten Mobilfunkgeneration und die damit verbundenen höheren Datenübertragungraten wurde die Anwendung von Multimedia Diensten möglich. Echtzeit-Dienste, wie zum Beispiel Videostreaming und Videotelephonie, stellen hierbei für Mobilfunksysteme eine besondere Herausforderung dar, vor allem wegen der zeitlich veränderlichen Qualität des Funkkanals. Das Videokompressionsverfahren benutzt eine Kombination aus zeitlicher und räumlicher Prädiktion, sowie transformierender und längenvariierender Kodierung. Dies erlaubt zwar eine höhere Datenkompression, jedoch steigt dadurch die Fehleranfälligkeit.

Diese Arbeit beschäftigt sich mit der Widerstandsfähigkeit der Videoübertragung über Mobilfunknetze. Das Hauptaugenmerk richtet sich auf den neuesten H264/AVC Standard, obwohl sich die vorgeschlagenen Methoden auch auf andere Videokodierungsstandards anwenden lassen. Der erste Teil der Arbeit beschäftigt sich mit dem Ausnutzen der Restredundanz des empfangenen Videostroms im Dekoder. Die Redundanz wird zur Fehlerlokalisierung in einem beschädigtem Datenpaket verwendet, das sonst verworfen werden würde. Fehlererkennung durch Syntaxanalyse und Artefakterkennung erfordert keine Erhöhung der Datenrate bei gleichzeitiger Reduktion der Verzerrung. Aufgrund der limitierten Erkennungsfähigkeit, werden auch zusätzliche Encoder-unterstützte Maßnahmen vorgeschlagen. Experimentelle Ergebnisse zeigen, dass speziell die Resynchronisierung des längenvariierenden Kodes, die Übertragung verbessert.

Im zweiten Teil der Arbeit werden Fehlerverdeckungsmethoden miteinander verglichen, verbessert und neu vorgeschlagen. Es gibt keine universelle Fehlerverdeckungsmethode, die optimale Ergebnisse in jeder Situation liefert. Die zeitliche Interpolation verhält sich im allgemeinen besser als die räumliche, außer im Falle eines Szenenwechsels. Deshalb wird ein adaptiver Mechanismus vorgeschlagen, der aus mehreren zeitlichen und räumlichen Methoden, je nach Situation, auswählt. Dazu gehört ein mit einer dynamischen Schwelle funktionierender Szenenwechseldetektor, der unahbhängig von Enkodereinstellungen arbeitet. Trotz seiner Einfachheit liefert die vorgeschlagene Methode eindrucksvolle Ergebnisse.

Der Schlußteil behandelt ein Cross-Layer-Design für UMTS. Es wird gezeigt, dass die Verwendung der Radio Link Control (RLC) Information in der Applikationsschicht eine klare Qualitätsverbesserung bringt ohne neue Kosten (kein Datenratenzuwachs, keine zusätzliche Komplexität) zu verursachen. Es wird außerdem gezeigt, dass Fehler in der RLC Schicht vorhersagbar sind. Die Unterscheidung zwischen Intervallen hoher und niedriger Fehlerwahrscheinlichkeit erlaubt die Zuordnung unterschiedlicher Prioritätsklassen zu den Videopaketen eines Benutzers. Ein Scheduler, der über solche Informationen verfügt, verbessert die Videoqualität erheblich.

# Acknowledgement

> *"The individual has always had to struggle to keep from being overwhelmed by the tribe. If you try it, you will be lonely often, and sometimes frightened. But no price is too high to pay for the privilege of owning yourself."*
>
> *F. Nietzsche*

THERE is a large number of colleagues and friends that substantially contributed to the success of my thesis.

I address particular thanks to my supervisor Prof. Dr. Markus Rupp, for providing me a challenging, instructive and highly positive PhD-experience. I greatly appreciate his full support and confidence in any situation and thank him for his interest and good discussions.

I would also like to thank mobilkom austria AG for technical and financial support of this work[1] as well as for their valuable feedback.

I would like to thank also my colleagues from the mobile communication group, for their research cooperation and support. Another thanks go to the colleagues from the Telecommunications Research Center Vienna (ftw.), in particular to Claudio Weidmann for his fruitful discussions.

Furthermore, I thank to all students I had the honor to supervise towards their bachelor and master theses, as well as to those that did their research projects with me. They helped me to broaden the field of my knowledge and experience.

Finally, I would like to thank my family, who made it possible for me to enjoy all my studies and always fully supported my educational plans. Special thanks, go to my husband Martin for his patience, understanding, and overall support.

Vienna, Mai 2007

*Olivia Nemethova*
e-mail: olivia@ieee.org

---

[1] The views expressed in this thesis are those of author and do not necessarily reflect the views within mobilkom austria AG.

# Contents

**1 Introduction 1**
- 1.1 Motivation . . . 2
- 1.2 Principles of H.264/AVC (Advanced Video Coding) . . . 3
  - 1.2.1 Video Sampling . . . 4
  - 1.2.2 Compression Mechanisms . . . 6
  - 1.2.3 Structure of Video Streams . . . 8
  - 1.2.4 Profiles and Levels . . . 9
  - 1.2.5 Reference Software . . . 10
- 1.3 H.264/AVC Video Streaming in Error-Prone Environment . . . 11
  - 1.3.1 Error Propagation . . . 11
  - 1.3.2 Standardized Error Resilience Techniques . . . 12
  - 1.3.3 Alternative Error Resilience Techniques . . . 13
- 1.4 Performance Indicators . . . 15
  - 1.4.1 Distortion . . . 15
  - 1.4.2 Data Rate . . . 16
  - 1.4.3 Computational Complexity . . . 17
- 1.5 Outline of the Thesis and Contributions . . . 17

**2 Error Detection 21**
- 2.1 Entropy Coding in H.264/AVC . . . 22
  - 2.1.1 Encoding . . . 22
  - 2.1.2 Consequence of Errors . . . 23
- 2.2 Error Detection by Syntax Analysis . . . 24
  - 2.2.1 Syntax Check Rules . . . 25
  - 2.2.2 Error Handling . . . 26
  - 2.2.3 Performance Evaluation . . . 27
- 2.3 Error Detection by Detection of Impairments . . . 30
  - 2.3.1 Detection in I Frames . . . 31
  - 2.3.2 Detection in P Frames . . . 33
- 2.4 Encoder Assisted Error Detection . . . 34
  - 2.4.1 Error Detection by Checksums . . . 34
  - 2.4.2 Watermarking for Error Detection . . . 35
- 2.5 VLC Resynchronization . . . 43
  - 2.5.1 Limitation of Distorted Area . . . 44
  - 2.5.2 Length Indicator Encoding . . . 45
  - 2.5.3 Performance Evaluation . . . 48
  - 2.5.4 Watermarking and VLC resynchronization . . . 49
- 2.6 Sequential Decoding . . . 49
  - 2.6.1 Sequential CAVLC Decoder . . . 50

2.6.2 Additional Synchronization Points . . . 52
2.6.3 Postprocessing . . . 53
2.6.4 Performance . . . 53

**3 Error Concealment** **57**
3.1 Spatial Error Concealment Methods . . . 58
3.1.1 Weighted Averaging . . . 59
3.1.2 Pixel Domain Directional Interpolation . . . 60
3.1.3 Maximally Smooth Recovery Methods . . . 63
3.1.4 Projection onto Convex Sets (POCS) . . . 69
3.2 Temporal Error Concealment Methods . . . 73
3.2.1 Motion Compensation . . . 74
3.2.2 Model Based Error Concealment . . . 77
3.3 Comparison of Individual Methods . . . 81
3.3.1 Quality of Reconstruction . . . 81
3.3.2 Possibility of Deployment . . . 82
3.3.3 Computational Complexity . . . 83
3.4 Adaptive Error Concealment Mechanism . . . 85
3.4.1 Scene Change Detection . . . 86
3.4.2 Decision Tree . . . 89
3.4.3 Performance Evaluation . . . 91

**4 Cross-Layer Design** **95**
4.1 Link Layer Aware Error Detection . . . 96
4.1.1 Error Detection at RLC Layer . . . 96
4.1.2 RLC PDU Based VLC Resynchronization . . . 97
4.1.3 Error Detection and VLC Resynchronization Efficiency . . . 99
4.2 Link Error Prediction Based Redundancy Control . . . 100
4.2.1 Link Error Prediction . . . 101
4.2.2 Redundancy Control . . . 102
4.3 Semantic Aware Scheduling . . . 105
4.3.1 Scheduling Mechanism . . . 106
4.3.2 Performance Evaluation . . . 108
4.4 Distortion Aware Scheduling . . . 112
4.4.1 Scheduling Mechanism . . . 112
4.4.2 Estimation of Intervals with Low Error Probability . . . 113
4.4.3 Distortion Estimation . . . 114
4.4.4 Performance Evaluation . . . 118

**5 Conclusions** **122**

**A Abbreviations and Symbols** **125**
A.1 List of Mathematical Symbols . . . 125
A.2 List of Abbreviations . . . 125

**B Test Video Sequences** **130**
B.1 Standard Video Sequences . . . 130
B.1.1 Foreman . . . 130
B.1.2 Carphone . . . 131
B.1.3 Akiyo . . . 131
B.1.4 Mobile . . . 131

B.1.5 Silent . . . 132
B.1.6 Glasgow . . . 132
B.2 Recorded Video Sequences . . . 133
B.2.1 Limbach . . . 133
B.2.2 Videoclip . . . 133
B.2.3 Squash . . . 134
B.2.4 Soccer . . . 134
B.2.5 Soccermatch . . . 134
B.3 Objective Measures of Sequence Characteristics . . . 135
B.3.1 Spatial and Temporal Information . . . 135
B.3.2 Comparison of Video Sequences . . . 136

**C UMTS Overview 138**
C.1 UMTS Architecture . . . 138
C.2 Quality of Service in UMTS . . . 140
C.3 Protocol Architecture . . . 141
C.3.1 Radio Interface Protocols . . . 141
C.3.2 Channels . . . 142
C.3.3 Packetization . . . 143

**D Channel Error Models 145**
D.1 Bit Error Models . . . 145
D.1.1 Binary Symmetrical Channel (BSC) . . . 145
D.1.2 Binary Input Additive White Gaussian Noise (BIAWGN) . . . 146
D.2 Link Layer Error Models . . . 146
D.2.1 Gilbert Model . . . 147
D.2.2 Improved Models . . . 147
D.2.3 Performance Comparison . . . 151
D.3 Transport Layer Error Model . . . 152
D.3.1 Memoryless Error Model . . . 152
D.3.2 Transport Layer Error Trace . . . 153
D.4 Prediction of Error Probability . . . 153

# Chapter 1

# Introduction

## Contents

**1.1 Motivation . . . . . 2**

**1.2 Principles of H.264/AVC (Advanced Video Coding) . . . . . 3**

1.2.1 Video Sampling . . . . . 4

1.2.2 Compression Mechanisms . . . . . 6

1.2.3 Structure of Video Streams . . . . . 8

1.2.4 Profiles and Levels . . . . . 9

1.2.5 Reference Software . . . . . 10

**1.3 H.264/AVC Video Streaming in Error-Prone Environment . . . . . 11**

1.3.1 Error Propagation . . . . . 11

1.3.2 Standardized Error Resilience Techniques . . . . . 12

1.3.3 Alternative Error Resilience Techniques . . . . . 13

**1.4 Performance Indicators . . . . . 15**

1.4.1 Distortion . . . . . 15

1.4.2 Data Rate . . . . . 16

1.4.3 Computational Complexity . . . . . 17

**1.5 Outline of the Thesis and Contributions . . . . . 17**

## 1.1 Motivation

AFTER the 2nd generation of mobile systems focusing on voice services, the 3rd generation was designed to enable multimedia communications. This required on the one hand higher data rates at the radio interface, accomplished mainly by adopting the Wideband Code Division Multiple Access (WCDMA) technology. On the other hand, a mechanism capable of controlling the Quality of Service (QoS) for different types of applications was indispensable. In Europe, the 3rd generation Universal Mobile Telecommunications System (UMTS) has been deployed since 2003 [1]. Its standardization had been carried out within the 3rd Generation Partnership Project (3GPP) [2] — a collaboration agreement established between several standardization and industrial bodies[1]. Updates of the UMTS standard organized in releases allow for its backward compatible enhancements, containing new features and converging contiguously to the next generation of mobile systems. The UMTS release 4 (implemented by the first UMTS network elements and terminals) provides a maximum data rate of 1920 kbit/s shared by all users in a cell, release 5 (emerging) offers up to 14.4 Mbit/s in downlink (DL) direction for High Speed Downlink Packet Access (HSDPA). The availability of such data rates initiated the launch of new services, out of which the real-time services are the most challenging from the provider point of view. The most popular among them are video streaming, video call and conferencing.

Video call is a two-way real-time service where two users communicate together by means of speech and video. This poses constraints on the end-to-end delay that should be as small as possible. The same applies to video conferencing that is a generalization of video calls to more than two users communicating together.

Video streaming is a one-way quasi real-time data transport, where the content is consumed (viewed/heared/read) while it is being delivered. To compensate jitter (variance of the end-to-end delay) at the receiver, a portion of the received data is buffered in a play-out buffer. In the case of video streaming, the video content is rendered on the screen with the signalized frame rate, making the inter-packet arrival time variations invisible for the user. Therefore, the end-user quality for video streaming does not depend on the absolute end-to-end delay (as long as it is kept in the order of seconds) but rather on the jitter. Thus, video streaming is usually referred to as a quasi real-time service, having delay constraints slightly relaxed compared to the real-time video call. However, if the jitter is greater than the play-out buffer size, data arriving later will be discarded — considered to be lost. In this way, the delay constraints transform into *packetloss*, the packetloss ratio reported over a certain time period. Throughout this work, not only the packets that did not arrive at the receiver within the required delay interval, but also the damaged packets (packets containing an error) are considered as lost. An entire data packet containing an error is typically discarded as well. Errors are detected by a checksum calculated over the whole packet. The checksum is capable of detecting the presence of error(s), but neither its/their position nor their amount. Discarding the entire packets although only one/few errors occurred, is highly inefficient especially if video is transmitted. Video, even if

---

[1] European Telecommunications Standards Institute (ETSI), Association of Radio Industries and Businesses / Telecommunication Technology Committee Japan (ARIB/TTC), China Communications Standards Association (CCSA), Alliance for Telecommunications Industry Solutions North America (ATIS) and Telecommunication Technology Association South Korea (TTA)

compressed, still contains redundancy that can be utilized by the decoder to determine more exactly the error position and, possibly, to recover a part of it. The picture areas that cannot be recovered by improved error detection must be concealed. The distortion of the decoded and concealed video depends not only on the packetloss ratio and error handling (detection, concealment), but also on the video content. The spatial and temporal correlation of the content determines the compression gain and the performance of the error concealment methods.

The packetloss ratio is given by the transport network and its configuration and conditions. In case of UMTS, the transport network is heterogeneous, it consists of several wireless and wired interfaces. The radio interface is the most critical part due to the high time variance of the wireless channel. The knowledge of the transported video packet characteristics in the network (content awareness) can help the network to prioritize and/or better protect the more important parts of video. On the other hand, the knowledge of the transport network (network awareness) at the encoder can help to adapt the encoding in a rate distortion optimal way. Both network and content awareness are only possible if the end-to-end video streaming architecture supports a cross-layer design, i.e., it allows for the exchange of information between the functionally separated protocol layers.

This thesis addresses error detection mechanisms, error concealment methods, as well as cross-layer design for video streaming. Efficient techniques for error resilient transmission of video over wireless networks are proposed, evaluated and compared. Nevertheless, many of the discussed techniques can be equally applied to video call and conferencing. The rest of this chapter provides a brief introduction to video coding, state-of-the-art error resilience techniques for transmission of video streams in error-prone environments, and a summary of indicators for the evaluation of the performance of such error resilience methods. An outline of the thesis and the author's contributions conclude the chapter.

## 1.2 Principles of H.264/AVC (Advanced Video Coding)

NOWADAYS for low-rate video streaming, the following video compression standards are used: H.263 [3] standardized by International Telecommunication Union (ITU), MPEG-4 part 2 [4] standardized by International Organization for Standardization (ISO) Motion Picture Expert Group (MPEG), and the emerging, newest and best performing H.264 [5] (known also as Advanced Video Coding (AVC) and MPEG-4 part 10), standardized by the Joint Video Team (JVT) of experts from both ISO/IEC (International Electrotechnical Commission) and ITU. The principle of the compression for all mentioned codecs is very similar.

In this thesis, the focus is given on H.264/AVC, especially on its baseline profile (cf. Section 1.2.4) designed for low complexity and low rate applications; the majority of analyzed experiments utilize an H.264/AVC codec. The H.264/AVC design covers a Video Coding Layer (VCL), designed to efficiently represent the video content, and the Network Abstraction Layer (NAL), which formats the VCL representation of the video and provides header information in a manner appropriate for conveyance by a variety of transport layers or storage media [6]. The output of NAL are NAL units (NALUs). This structure is visualized in **Figure 1.1**. Data Partitioning (DP) is an optional part, explained in Section 1.3.2.

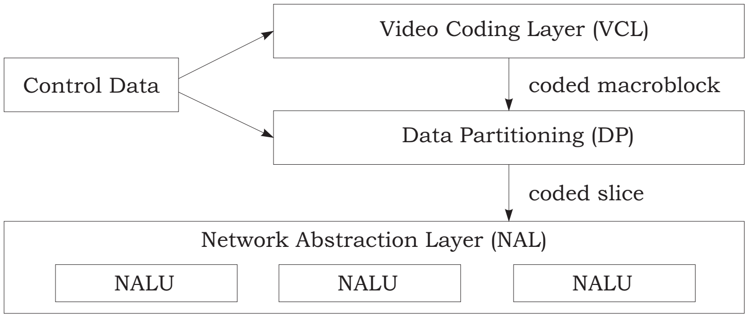


**Figure 1.1**: Layer structure of H.264/AVC encoder.

### 1.2.1 Video Sampling

Digital video cameras perform two kinds of sampling — sampling in the temporal domain given by the number of pictures per second (frame rate), and sampling in the spatial domain given by the number of points (pixels) in each of the pictures (picture resolution).

*Frame rate*, or frame frequency, is the measure of how quickly an imaging device produces unique consecutive images called frames. The frame rate is most often expressed in frames per second (f/s) or alternatively, in Hertz (Hz). The today's common frame rates have their origins in analog television systems. National Television Systems Committee (NTSC) standardized the NTSC system, which is in use in Canada, Japan, South Korea, USA, and some other places in south America, working with 29.97 f/s (denoted commonly as 30 f/s). In the rest of the world Phase Alternation by Line (PAL) and Séquentiel couleur à mémoire (SECAM) are used, operating at a frame rate of 25 f/s. In some low-rate and low-resolution applications the frame rate is reduced before the actual transmission to save data rate. Frame rate reduction can be performed by decimating the frame rate by $F$ — leaving each $F$th frame while removing the rest. A typical example is mobile video streaming or call/conferencing with usual frame rates decimated by 2,3,4 or even 5. Other frame rates can be obtained by interpolation and subsequent decimation.

Furthermore, video cameras can work with two different image capture formats: *interlaced scan* and *progressive scan*. Interlaced cameras record the image in alternating sets of lines: the odd-numbered and the even-numbered lines. One set of odd or even lines is referred to as a *field*, and a consecutive pairing of two fields of opposite parity is called a frame. A progressive scanning digital video camera records each frame as distinct, with both fields being identical. Thus, interlaced video captures twice as many fields per second as progressive video does when both operate at the same number of frames per second. In contrast to televisions, computer monitors generally use progressive scan, and therefore internet/mobile video formats use it, too. H.264/AVC supports both interlaced and progressive scan.

Each frame consists of *pixels*. Pixels of intensity pictures (black-and-white) are scalar values; pixels of color pictures are represented by coordinates within the relevant color space. The captured RGB picture is thus represented by three $N \times M$ color component matrices consisting of $q$-bit long numbers (usually $q = 8$). In **Table 1.1** some resolutions common for internet and mobile video are summarized. Since the human visual system is less sensitive to color than to luminance (brightness), bandwidth can be optimized by storing more luminance detail than color detail. At normal viewing distances,

| Abbreviation | Size | Description |
|---|---|---|
| VGA | 640×480 | Video Graphics Array |
| QVGA | 320×240 | Quarter Video Graphics Array,<br>called also Standard Interchange Format (SIF) |
| Q2VGA | 160×120 | |
| CIF | 352×288 | Common Intermediate Format<br>(quarter of resolution 704×576 used in PAL) |
| QCIF | 176×144 | Quarter Common Intermediate Format |

**Table 1.1**: Typical picture resolutions in pixels, used for internet and mobile video applications.

there is no perceptible loss incurred by sampling color details at a lower rate. In video systems, this is achieved by using the color difference components. The signal is divided into a luminance (denoted as Y, called shortly 'luma') and two color difference (chrominance) components, denoted as U and V (or Cb and Cr, respectively), called shortly 'chroma'.

The YUV signals are created from an original RGB (red, green and blue) source as follows [7]. The weighted values of R, G and B are added together to produce a single Y signal, representing the overall brightness, or luminance, of that spot:

$$Y = k_r \cdot R + k_g \cdot G + k_b \cdot B, \tag{1.1}$$

where $k_r, k_g$ and $k_b$ are weighting factors, $k_b + k_r + k_g = 1$. ITU-R recommendation BT.601 [8] defines $k_b = 0.114$ and $k_r = 0.299$.

The U signal is then created by subtracting Y from the blue signal of the original RGB, and a scaling operation; and V by subtracting Y from the red, and then scaling by a different factor. The following formulas approximate the conversion between the RGB color space and YUV.

$$\begin{aligned} Y &= k_r \cdot R + (1 - k_b - k_r) \cdot G + k_b B, \\ U &= \frac{0.5}{1 - k_b} \cdot (B - Y), \\ V &= \frac{0.5}{1 - k_r} \cdot (R - Y). \end{aligned} \tag{1.2}$$

The basic idea behind the YUV format is that the human visual system is less sensitive to high frequency color information (compared to luminance) so the color information can be encoded at a lower spatial resolution. The most common way of subsampling, called 4:2:0 reduces the number of samples in both the horizontal and vertical dimensions by a factor of two, i.e., for four luma pixels there is only one blue and one red chroma pixel. Hence, the YUV color space subsampling is the first step to data rate reduction. The original bit rate $R_{\text{raw}}$ of the *raw* (uncompressed) RGB video with frame rate $f_r$ and picture resolution $M \times N$ is given by $R_{\text{raw\_RGB}} = 3 \cdot r_f \cdot M \cdot N \cdot q$; the corresponding raw YUV video only requires rate $R_{\text{raw\_YUV}} = 1.5 \cdot r_f \cdot M \cdot N \cdot q$. For the QCIF resolution raw video with 25 f/s and $q = 8$, the necessary bit rate is $R_{\text{raw\_YUV\_QCIF}} = 7.6$ Mbit/s. Although the bit rate is reduced to 50% of the RGB raw video, it is still not feasible for any today's internet or mobile application. To overcome this, compression is further employed to reduce the data rate as described in the following sections.

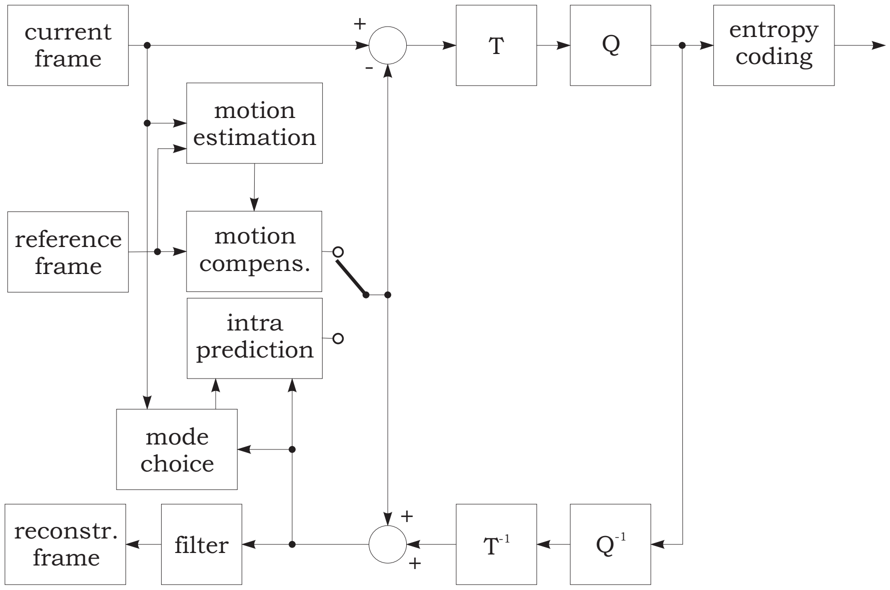


**Figure 1.2**: Dataflow diagram of H.264/AVC encoder.

### 1.2.2 Compression Mechanisms

The video coding layer of H.264/AVC consists of a hybrid of temporal and spatial prediction, together with transform coding. Its dataflow diagram is depicted in **Figure 1.2**. Each frame is split into non-overlapping areas — *macroblocks* (MB) — consisting of 16×16 samples of the luma and 8×8 samples of each of the two chroma components. The macroblocks are organized in *slices*, representing subsets of macroblocks that can be decoded independently.

Frames are called *intra-coded* if they are encoded by means of a *spatial prediction* without using information other than that contained in the picture itself. Typically, the first picture of a video sequence is intra-coded as well as all random access point of the video sequence (pictures that can be fast accessed without decoding previous parts of video sequentially). Each macroblock in an intra-coded frame (called also intra-frame or I frame) is predicted using spatially neighboring samples of previously coded macroblocks[2]. The encoder performs a mode choice — it decides which and how neighboring samples are used for intra prediction. The chosen intra prediction type is then signalized within the bitstream.

For all remaining pictures of a sequence between random access points, typically *inter-coding* is used, employing temporal prediction from other previously decoded pictures. First, the *motion estimation* of each block is performed by searching the best matching region from the previous or following frame(s). Note that the best match is not searched for the original (uncompressed) block, but rather for the quantized and filtered block. This prevents artifacts during the reconstruction process. The best match is taken as a prediction of the encoded block. Such a prediction is thus called *motion compensated*. Each inter-coded macroblock is subject to further partitioning into fixed-size

---

[2] Macroblocks that do not have any previously encoded neighbors (e.g. the first MB in picture and MBs at the top slice boundary) are encoded without prediction.

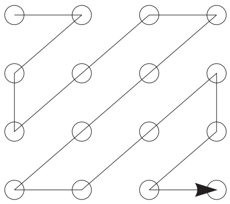


**Figure 1.3**: Scanning of samples in a submacroblock.

blocks (16×16 luma samples corresponding to no partitioning, 16×8, 8×16 or 8×8) used for motion description. Blocks of size 8×8 can be split again into submacroblocks (SMB) of 8×4, 4×8, or 4×4 luma samples. Chrominance parts are segmented correspondingly.

H.264/AVC supports multi-picture motion compensated prediction — more than one previously coded picture can be used as a reference. The accuracy of motion compensation is a quarter of a sample distance. The prediction values at half-sample positions are obtained by applying a one-dimensional six tap Finite Impulse Response (FIR) filter. Prediction values at quarter-sample positions are generated by averaging samples at integer- and half-sample positions. To enable unambiguous reconstruction at the receiver, the *motion vector* (MV) between the position of the block within the frame and the position of its best match in the previously encoded frame has to be signalled as well as the mode of segmentation and corresponding reference frame(s). To avoid signalizing of the zero motion vectors and zero residuals in the cases of static picture parts, the *SKIP mode* allows for skipping of signalized number of P/B macroblocks. In SKIP mode neither residuals, nor motion vectors are sent. At the receiver, the spatially corresponding macroblock from the previous frame is taken.

Inter-coded frames are referred to as inter-frames or P and B frames; P being the frames which use for prediction only previous frames, B being the *bi-directionally predicted* frames that use for prediction also successive frames. In H.264/AVC, other pictures can reference B frames for the motion estimation. The substantial difference between P and B macroblocks is that B MBs may use a weighted average of two distinct motion-compensated prediction values for building the prediction signal. Note that H.264/AVC supports frames with mixed I,P and B slices. Moreover, P and B slices may contain some I MBs.

All luma and chroma samples of an MB are either spatially or temporally predicted and the resulting *prediction residuals* (difference between the macroblock samples being encoded and their prediction) are transformed. H.264/AVC uses three transforms depending on the type of residual data that is to be coded: a Hadamard transform for the 4×4 array of luma DC (direct current) coefficients in I MBs predicted in 16×16 mode, a Hadamard transform for the 2×2 array of chroma DC coefficients (in any type of MB), and a 4×4 integer Discrete Cosine Transformation (DCT) for all other blocks. The result of the transformation is a matrix of *coefficients* corresponding to different spatial frequencies. The coefficient corresponding to the lowest frequency is DC, the others are denoted AC (alternating current) coefficients. All coefficients are further quantized. For each macroblock the quantization is controlled by the Quantization Parameter (QP) ranging from zero to 52. The quantization indices are scanned in zig-zag order and finally entropy encoded as described in Section 2.1, together with other signalling information. The zig-zag order used in H.264/AVC is illustrated in **Figure 1.3**.

In H.264/AVC, a macroblock can be coded in one of the many possible modes that are enabled depending on the picture/slice type. The *mode decision* is performed at the encoder, i.e., it is not within the scope of a standard[3]. Additional important gains in coding efficiency become possible if a macroblock mode decision is performed carefully. However, the additional gains can be extracted only at the expense of considerable increase in encoding complexity (e.g. by implementing a rate-distortion optimization (RDO) at the encoder).

If the encoder parameters (QP, MV search area, etc.) are kept during the encoding, then the number of coded bits produced for each macroblock will change depending on the content of the video frame and the mode decision, causing the bit rate of the encoder to vary. This variation in bit rate can cause problems especially in systems where resources are shared (e.g. wireless systems), the resource management cannot efficiently perform the resource allocation. The Variable Bit Rate (VBR) produced by an encoder can be smoothed by buffering the encoded data prior to transmission in a FIFO (First-In/First-Out) buffer, which is emptied at a Constant Bit Rate (CBR) matched to the channel capacity. Another FIFO buffer is placed at the input to the decoder and is filled at the channel bit rate and emptied by the decoder at a variable bit rate (since the number of bits to be extracted per frame is varying over frames, but still the frames have to be rendered on the display with a constant frame rate). However, the costs are the buffer storage capacity and delay — the wider the bit rate variation, the larger the buffer size and decoding delay. Another possibility to compensate the VBR is the *rate control* using the quantizer adaptation. However, quantizer changes need to be carefully restricted based on scene complexity, picture types, and coding bit rate to maintain an acceptable end-user quality. H.264/AVC allows for an individual quantization parameter per macroblock, which can introduce spatial quality variations in fine texture.

An unpleasant effect of the block based coding is blocking (also known as blockiness), the visible block structures in the compressed picture. It is a consequence of the quantization. Blocking was shown to be one of the most annoying artifacts resulting from the lossy compression — the one which reduces the quality experienced by user (subjective quality) at most [9]. To provide better subjective quality, H.264/AVC defines an adaptive *in-loop deblocking filter*, where the strength of filtering is controlled by the values of several syntax elements [10]. The blockiness is reduced without much affecting the sharpness of the content. Consequently, the subjective quality is significantly improved. Moreover, the filter reduces the bit-rate by typically 5-10% while producing the same quality as the non-filtered video.

### 1.2.3 Structure of Video Streams

The network abstraction layer formats the compressed video data coming from the video coding layer and provides additional non-VCL information, such as sequence and picture parameters, access unit delimiter, filler data, Supplemental Enhancement Information (SEI), display parameters, picture timing etc., in a way most appropriate for a particular network. All data related to a video stream are encapsulated in NAL Units (NALU). The format of a NALU is shown in **Figure 1.4**. The first byte of each NALU is a header byte, the rest is the data. The first bit of the NALU header is a zero (forbidden) bit. The following two NRI (NAL Reference Identification) bits signalize the importance

---

[3] Video codec standards define the functions and features of the decoder rather than encoder.

| | | | |
|---|---|---|---|
| 0 | NRI | NALU type | NALU payload |

**Figure 1.4**: Format of a NALU and its header.

of the NAL unit for reconstruction purposes. The next five bits indicate the NALU type corresponding to the type of data being carried in that NALU. There are 32 types of NALUs allowed. These are classified in two categories: VCL NALUs and non-VCL NALUs. The NALU types from one to five are VCL NALUs and contain data related to the output of VCL — slices. Each encoded slice is also attached a header containing information related to that slice.

NALUs with a NALU type indicator value higher than five are non-VCL NALUs carrying information like SEI, sequence and picture parameter set, access unit delimiter etc. Depending on a particular delivery system and scheme, some non-VCL NALUs may or may not be present in the stream containing VCL NALUs. For example, NALU type seven carries the Sequence Parameter Set (SPS), defining profile, resolution and other overall properties of the whole sequence; type eight carries the Picture Parameter Set (PPS), containing type of entropy coding, slice group and quantization properties. These sequence and picture level data can be sent asynchronously and in advance of the media stream contained in the VCL NALUs. An active SPS remains unchanged troughout a coded video sequence. An active PPS remains unchanged within a coded picture. In order to be able to change picture parameters such as picture size without the need to transmit parameter set updates synchronously to the slice packet stream, the encoder and decoder can maintain a list of more than one SPS and PPS. Each slice header contains then a codeword that indicates the SPS and PPS in use.

The NALUs can easily be encapsulated into different transport protocols and file formats, such as MPEG-2 transport stream, RTP (Real-Time Protocol) and MPEG-4 file format. For transmission over mobile networks, a VCL slice is encapsulated in RTP according to [11]. The RTP payload specification supports different packetization modes. In the simplest mode a single NALU is transported in a single RTP packet, and the NALU header serves as an RTP payload header. In non-interleaved mode, several NALUs of the same picture can be encapsulated into the same RTP packet. In interleaved mode several NALUs belonging to different pictures can be encapsulated into the same RTP packet. Moreover, NALUs do not have to be sent in their decoding order. Both the non-interleaved and interleaved modes also allow for fragmentation of a single NALU into several RTP packets. The further mapping of such encoded video packets on the protocol layers of UMTS can be found in Appendix C.3.

### 1.2.4 Profiles and Levels

Since H.264/AVC is designed for a multitude of applications, the structure of its bitstream may vary significantly. To avoid implementing of all possible stream structures by each specification-conform decoder, *profiles* and *levels* were defined. A profile is a subset of the capabilities including the entire bitstream syntax; a level is a specified set of constraints imposed on values of the syntax elements in the bitstream. Levels allow for standard-compliant low-complexity encoder and decoder implementations.

At present, H.264/AVC standard [5] includes the following seven profiles, targeting specific classes of applications:

- **Baseline Profile**: Primarily for lower-cost applications demanding less computing resources, this profile is used widely in video conferencing and mobile applications.

- **Main Profile**: Originally intended as the mainstream consumer profile for broadcast and storage applications, the importance of this profile faded when the High profile was developed for those applications.

- **Extended Profile**: Intended as the streaming video profile, this profile has relatively high compression capability and some extra tricks for robustness to data losses and server stream switching.

- **High Profile**: The primary profile for broadcast and disc storage applications, particularly for high-definition television applications (this is the profile adopted into HD DVD and Blu-ray Disc, for example).

- **High 10 Profile**: Going beyond today's mainstream consumer product capabilities, this profile builds on top of the High Profile — adding support for up to 10 bits per sample of decoded picture precision.

- **High 4:2:2 Profile**: Primarily targeting professional applications that use interlaced video, this profile builds on top of the High 10 Profile — adding support for the 4:2:2 chroma sampling format while using up to 10 bits per sample of decoded picture precision.

- **High 4:4:4 Profile**: This profile is essentially similar to High 4:2:2 Profile. Additionally, it supports up to 4:4:4 chroma sampling, up to 12 bits per sample, and efficient lossless region coding and integer residual color transform for coding RGB video while avoiding color-space transformation error.

In this thesis, focus is given on the baseline profile. Note that baseline profile does not support B slices, only I and P slices are possible. Other baseline profile constraints will be discussed later, when necessary for the application.

### 1.2.5 Reference Software

All experiments with H.264/AVC codecs presented in this work, were performed using the Joint Model (JM) reference software [12] developed by JVT for testing. JM includes both the encoder and the decoder compliant with the H.264/AVC hypothetical reference decoder [13]. Settings are passed via command line and/or the configuration files `encoder.cfg` and `decoder.cfg`. JM implements all standardized features as well as some basic non-standardized methods e.g. RDO, rate control, error concealment. However, there still are (in the time of writing the thesis) some bugs in this reference software, e.g. the error concealment does not work properly and causes the crash of the decoder.

For the aims of this thesis, the encoder and decoder standardized functions were not modified, rather several new features and interfaces were added as described later on.

## 1.3 H.264/AVC Video Streaming in Error-Prone Environment

THIS section describes the effect of errors on the decoding of an H.264/AVC video stream. It briefly summarizes standardized error resilience means, presents open problems and summarizes the most popular frameworks for additional error resilience according to the state of the art.

The usual way of transport of video streaming over IP (Internet Protocol) packet networks assumes an RTP together with Real-Time Control Protocol (RTCP) feedback on the application/session layer and a User Datagram Protocol (UDP) on the transport layer [14], [15]. In contrary to the Transmission Control Protocol (TCP), UDP does not provide any Automatic Repeat Request (ARQ) mechanism to perform retransmissions. It only provides a checksum to detect possible errors. The checksum is typically calculated over a rather large packet to avoid a rate increase due to packet headers.

The interactive and background services having a non-real-time nature (e.g. web browsing, file transfer, e-mail) can make use of retransmission mechanisms, e.g. provided by TCP — packetloss is compensated by delay. This is completely different for the real-time conversational and streaming services. The packetloss caused by limited delay does not necessarily reflect the differences in the residual bit error distribution — the distribution of bit errors within the packets containing an error. If the number of residual bit errors is low, a decoding of such stream might even result in lower distortion than using error concealment (cf. Section 2.2). This section describes the impact of errors in H.264/AVC on the distortion of decoded picture and presents some standardized techniques that aim on its reduction. Finally, the most popular alternative state-of-the-art error resilience techniques are summarized.

### 1.3.1 Error Propagation

The compression mechanism introduced in the previous section is capable of reducing the data rate more than 100 times while keeping the user perceived quality almost unchanged. It has however some drawbacks to the robustness of such encoded video stream against transmission errors. An error in a single bit of stream may cause considerable distortion due to the error propagation.

There are three possible sources of error propagation in an H.264/AVC encoded video stream: entropy code, spatial prediction, and temporal prediction. Entropy codes assign codewords to symbols to match codeword lengths with probabilities of the symbols — entropy codes are variable length codes (VLC). Hence, an error in a codeword may impact following codewords as well, if the codeword boundaries are determined incorrectly. This is also a reason for discarding the entire packet if an error is detected. A detailed analysis of the H.264/AVC entropy code and impact of errors on it is presented in Section 2.1.

Spatially (intra) encoded MBs require for decoding the spatially neighboring reference blocks. If these are erroneous, the MB reconstructed using them will be distorted, too. In practice, however, this problem does not occur as far as the entire packet is discarded if an error was detected. However, spatial error propagation becomes significant for the techniques utilizing an information from damaged packets.

Temporal (inter) prediction causes propagation of errors between the frames of the same video sequence. If an error occurs in a frame, all following frames that use such frame as a reference for the motion compensation will be reconstructed erroneously. The information in the video sequence is refreshed in each inter frame by the new information contained in the transmitted residuals, thus the sequences containing a higher amount of movement recover faster. Another sources of refreshing are the I frames and the intra-coded MBs inserted (randomly) in inter-predicted slices.

The effects of spatial and temporal error propagation are illustrated in **Figure 1.5**. The green color marks the impacted area.

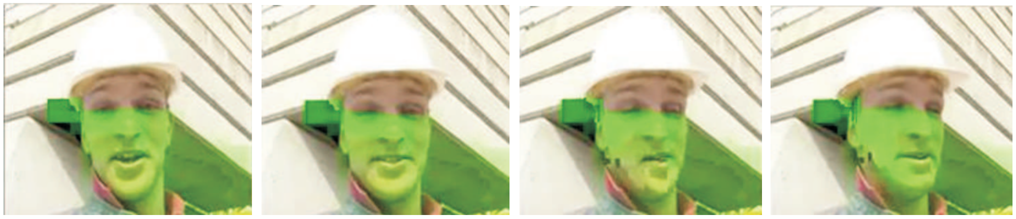


**Figure 1.5**: Example of the spatial and temporal error propagation in a H.264/AVC encoded video sequence “foreman”.

### 1.3.2 Standardized Error Resilience Techniques

To reduce the distortion resulting from the errors and their propagation, several means were added to the H.264 standard. They are summarized nicely e.g. in [16] and [17]. In [18], the benefits of the H.264/AVC error resilience features in the context of wireless transmission are emphasized. In the following, the most important of these features are summarized.

- **Slices** provide spatially distinct resynchronization points within the video data for a single frame. This is accomplished by introducing a slice header containing syntactical and semantical resynchronization information. Intra prediction and motion vector prediction are not allowed over slice boundaries. A trivial slicing method is to use one frame corresponding to one slice. The other possibility is the subdivision of the frame into slices with a fixed maximum number of MBs, or slices with a fixed maximum number of bytes.

- **Flexible Macroblock Ordering** (FMO) allows for an arbitrary mapping of MBs to slice groups by means of macroblock allocation maps.

- **Arbitrary Slice Ordering** (ASO) allows the decoding order of slices within a picture not to follow the constraint that the address of the first MB within a slice is monotonically increasing within the NALU stream for a picture. This permits e.g. reduction of decoding delay in case of out-of-order delivery of NALUs.

- **Intra refreshing** helps to stop the temporal error propagation. Apart from the regular sending of the whole intra-coded frames connected with high peaks in rate complicating resource allocation, H.264/AVC allows for sending intra-slices and even intra-macroblocks within the inter-frames.

- **Redundant slices** (RS) are an error resilience feature allowing an encoder to send an extra representation of a picture region (typically at lower fidelity) that can be used if the primary representation is corrupted or lost.

- **Data Partitioning** (DP) provides the ability to separate more important and less important syntax elements into different packets of data, enabling the application of unequal error protection (UEP) and other types of improvement of error/loss robustness. Header data and motion vectors are labeled *type A*, so that they can be better protected. The residuals of intra frames are labeled *type B*, while inter-predicted residuals are *type C*.

Note that these error resilience techniques can be combined to achieve the performance required for particular applications and scenarios. All above mentioned error resilience features except DP are available in the baseline profile. DP is available only in the extended profile.

### 1.3.3 Alternative Error Resilience Techniques

Apart from techniques standardized by H.264/AVC, there are other possibilities to enhance error resilience of the video transmission:

- techniques considered for future standard improvements,

- methods that were considered but not selected in the final standard,

- implementation specific mechanisms that are not in the scope of standardization,

- techniques that consider more layers/standards optimizing them jointly.

The most popular state-of-the-art frameworks and terms for error resilient video transmission are summarized in this section.

For H.264/AVC two extensions are considered and worked on: multi-view coding and *scalable video coding* (SVC), the latter being a mean for error resilient video transmission [19]. The SVC design can be classified as *layered video coding*. A scalable representation of video consists of a base layer (providing a basic quality level) and multiple enhancement layers (serving as a refinement of the base layer, not usefull without base layer). H.264/SVC also supports three types of inter-layer prediction (motion, intra, residual). The spatial and temporal resolution scalability is provided at a bit-stream level, i.e., by simply discarding the corresponding NALUs. The benefit of H.264/SVC for wireless multiuser streaming is demonstrated in [20].

*Multiple description coding* (MDC) is an alternative to the layered coding concept. In contrary to layered coding, MDC [21] generates a number of equivalent-importance data streams that, all together, carry the input information. Sending of RS in H.264/AVC may also be understood as a form of MDC. So far, various MDC schemes were proposed and evaluated in literature for transmission over error prone environments, e.g. [22]. It has been argued whether layers or versions provide better means for multiuser error resilient transmission in lossy networks. The analysis in [23] and [24] demonstrated that layered coding schemes perform well in the cases where the packet transmission schedules can be optimized in a rate-distortion (RD) sense. For the scenarios with non RD-optimized packet schedules, with somewhat relaxed constraints on aggregate rate[4], MDC provides better results.

Of course, both concepts — layered coding and MDC — can be advantageously combined also with *unequal error protection* (UEP) in the lower layers. A basis for the UEP concept is the categorization of application layer data according to their importance. The importance of data is given by the distortion their loss would cause. The categorization can be coarse, typically based on semantic information, or finer, based on distortion estimation. Distinguishing between intra and inter coded frames, and prioritizing the intra coded packets is a typical simple *semantic-based* approach that can be found e.g. in [25] and [26]. A finer, *content-based* approach can be seen e.g. in [28], where encoder applies FMO according to the calculated distortion and protects unequally different slice groups. The protection itself can also have different forms. Additional protection on the application layer can be used as e.g. in [28], where Reed-Solomon code is employed. Alternatively, the mechanism may rely on the (channel) codes of the lower layers, or prioritize important packets in the scheduler as e.g. in [29].

A step forward from adapting the channel code for a given source is the joint design of those two. The well-known Shannon *separation theorem* [30] states that an optimal communications system can be constructed by considering separately the source and channel coder designs. It assumes, however, that the source code is optimal, i.e., removes all source redundancy. Moreover, it considers that for rates below channel capacity, the channel coder corrects all errors. These assumption do not really hold in practical systems with limited delay and complexity. Here, properly designed *joint source-channel coding* (JSCC) is more beneficial. The JSCC term includes many different concepts (a concise review is provided e.g. [31]) ranging from the channel-optimized (vector) quantization [32], decoding exploiting the residual redundancy [33],[34] up to solutions using overcomplete frame expansions [35]. Layered coding and MDC in combination with UEP and/or RD-optimized can also be considered as JSCC.

Clearly, an appropriate design of source coding alone may also improve the robustness of transported streams against errors. Therefore, various *reversible VLC codes* were proposed so far (e.g. [36], [37]) that prevent desynchronization of the VLC decoding. Alternatively, *resynchronization* information may be embedded within the video stream as e.g. in [38],[39]. All these mechanism result in a rate increase. The rate increase is justified as far as the distortion for the same rate decreases. This, however, depends often on the channel characteristics and conditions. The choice of the appropriate method can thus be seen as a *rate-distortion optimization* (RDO) problem. Typically, solving of rate-distortion problems is rather complex. On the other hand it leads to efficient resource usage [40].

There have been a lot of research effort on RD-optimized scheduling methods of packets over lossy networks (e.g. [41], [42], [43]). In heterogeneous mobile communication systems like UMTS, not each network element implements all protocol layers. Thus, the scheduler typically does not have information about the content transported. Such information is, however, essential for knowledge/estimation of the distortion. Hence, the strict protocol layer structure has to be evaded. In general, techniques that disregard the protocol layer structure and enable exchange of information and/or joint optimization at more protocol layers are known as *cross-layer design.* In fact, most of JSCC and UEP approaches can be considered as cross-layer.

---

[4] the total rate — sum of the rates of all streams sharing the same constraint resources.

Open System Interconnection (OSI) model [44] defines seven layers distinguished by their functionality. The basic idea of layering is that each layer adds value to services provided by the set of lower layers. Thus, layering divides the total problem into smaller pieces, that can be designed separately, which on the one hand considerably reduces the problem solving complexity, on the other hand, it limits the information available to a particular layer, which can lead to suboptimal solutions. Lately, especially for wireless communications, the cross-layer approaches gained on importance [45], [46], [47].

Finally, error concealment techniques determine the error "visibility" in the decoded video stream and belong thus to error resilience mechanisms. In [48] a summary of some well-known methods is presented; apart from that, there is a vast of literature proposing particular error concealment methods for various systems. The performance of error concealment depends strongly on instantaneous spatial and temporal characteristics of the video sequence as well as on the error pattern to be concealed. Thus, more than a single error concealment method, an adaptive mechanism is needed to better exploit the remaining spatial and temporal correlation of the decoded video stream [49]. Furthermore, for power-limited mobile devices and real-time applications it is essential to keep the complexity as low as possible.

## 1.4 Performance Indicators

THE performance of error resilience methods can be evaluated by measuring the end-user distortion in the presence of transmission errors with given characteristics and by analyzing the cost determined by increasing the rate and/or complexity. In this section, metrics are presented that are used throughout this work for performance evaluation.

### 1.4.1 Distortion

To evaluate the distortion within the $n$th video frame $\underline{\underline{F}}_n$ with respect to the reference (distortion-free) frame $\underline{\underline{R}}_n$, the mean square error (MSE) can be used:

$$\mathrm{MSE}[n] = \frac{1}{M \cdot N \cdot |\mathcal{C}|} \sum_{c \in \mathcal{C}} \sum_{i=1}^{N} \sum_{j=1}^{M} [\underline{\underline{F}}_n^{(c)}(i,j) - \underline{\underline{R}}_n^{(c)}(i,j)]^2, \tag{1.3}$$

where $N \times M$ is the size of the frame and $\mathcal{C}$ is the set of color components, e.g. for RGB color space $\mathcal{C} = \{R, G, B\}$. The number of color components corresponds to the cardinality $|\mathcal{C}|$ of the set. Row index $i$ and column index $j$ address the individual elements of the color component matrix.

For image distortion, the MSE is most commonly quoted in terms of the equivalent reciprocal measure Peak Signal to Noise Ratio (PSNR) defined for one picture/frame as:

$$\mathrm{PSNR}[n] = 10 \cdot \log_{10} \frac{(2^q - 1)^2}{\mathrm{MSE}[n]} \text{ [dB]}, \tag{1.4}$$

where $q$ represents the number of bits used to express the color component values. In YUV color space the two chrominance parts are usually smoother than the luminance. Therefore, especially for the applications that handle all components in the same way, it may be more discriminative to compare the distortion of luminance only ($\mathcal{C} = \{Y\}$) by means of Y-MSE$[n]$ or Y-PSNR$[n]$.

The PSNR averaged over the whole video sequence can be defined in two ways — as an average $\overline{\mathrm{PSNR}}$ over the PSNR$[n]$ of all its frames, or as

$$\mathrm{PSNR} = 10 \cdot \log_{10} \frac{(2^q - 1)^2}{\overline{\mathrm{MSE}}} \text{ [dB]}, \tag{1.5}$$

where $\overline{\mathrm{MSE}}$ is average MSE defined by

$$\overline{\mathrm{MSE}} = \frac{1}{F} \sum_{n=1}^{F} \mathrm{MSE}[n] \tag{1.6}$$

and $F$ is the number of frames in the video sequence.

The later definition is formally correct, since it averages over the linear values. Averaging over logarithmic values results in a systematic error as a consequence of Jensen's inequality [50], which states that if $f(\cdot)$ is a convex function and $X$ is a random variable, then

$$\mathrm{E}\{f(X)\} \geq f(\mathrm{E}\{X\}), \tag{1.7}$$

where $\mathrm{E}\{X\}$ denotes the expectation of $X$. A function $f(\cdot)$ is convex over an interval $(a, b)$ if for all $x_1, x_2 \in (a, b)$ and $0 \leq \lambda \leq 1$ applies

$$f(\lambda \cdot x_1 + (1 - \lambda) \cdot x_2) \leq \lambda \cdot f(x_1) + (1 - \lambda) \cdot f(x_2), \tag{1.8}$$

i.e., all points of $f(x)$ between the arbitrary chosen $x_1, x_2 \in (a, b)$ lay below the line connecting them.

Since $\log_{10}(1/x) = -\log_{10}(x)$ is convex, $\overline{\mathrm{PSNR}}$ will in general provide "better" results (meaning higher quality) than PSNR. In practice and in literature, both ways of calculating PSNR can be found. For instance, JM outputs the $\overline{\mathrm{PSNR}}$ value. Thus, in this thesis, the Y-PSNR values averaged per sequence correspond also to $\overline{\mathrm{PSNR}}$ in order to ease the comparison with literature where the JM experiments are presented.

Whereas PSNR estimates the user perceptual subjective quality well for packetloss impairments, it does not necessarily reflect the user evaluation of compression impairments. Nevertheless, it is still the mostly used distortion metric and therefore also one that allows for comparing the results with other methods without having to implement them. The author's contributions to estimation of subjective video quality can be found in [51]–[60].

### 1.4.2 Data Rate

After compression, the video sequence is represented with a corresponding string of binary digits (bits) denoted by $\underline{b}$. The objective is to keep its length $b$ as small as possible.

The average bit rate $R_b$ is the average number of bits used per unit of time to represent a continuous medium such as audio or video. It is quantified in bits per second (bit/s)

$$R_b = \frac{b}{T} \text{ [bit/s]}, \tag{1.9}$$

where $T$ is the time needed for the transmission of $\underline{b}$ from sender to receiver. The rate increase $\Delta R$ will be used to quantify the change in rate caused by a new method resulting in the rate $R_{\text{new}}$ compared to a reference rate $R_{\text{ref}}$

$$\Delta R = \frac{R_{\text{new}} - R_{\text{ref}}}{R_{\text{ref}}} \cdot 100 = \frac{b_{\text{new}} - b_{\text{ref}}}{b_{\text{ref}}} \cdot 100 \ [\%], \tag{1.10}$$

under the assumption that $T$ has to be the same for the reference bitstream and the bitstream after the application of the new method. However, for some applications (e.g. resource allocation) the average bit rate is not as important as the peak bit rate. The frame rate determines the time of the picture rendering. The play-out buffer compensates partially for the variable bit rate.

### 1.4.3 Computational Complexity

Computational complexity is usually measured by means of computational resources required to solve a given computational problem. Basic computational resources are number of steps (operations) necessary to solve a problem, memory space, the amount of storage needed while solving the problem and computational time or some more complicated resources. The amount of resources necessary for the implementation of a method depends, however, on the implementation itself. The optimization of the implementation is out of the scope of this thesis. Therefore, in this work an estimate of the number of basic arithmetical operations (additions/multiplications) will be used for simple methods. Note that the number of necessary operations can also vary for different implementation depending on the design goal, e.g. if spending memory or processor power is preferred.

## 1.5 Outline of the Thesis and Contributions

THIS thesis focuses on error resilient transmissions of video over wireless networks. It is nearly impossible to encompass and review all approaches that can improve error resilience. Hence, three promising directions were selected to be investigated in this thesis: error detection, error concealment, and cross-layer design. All three topics are discussed in the context of wireless transmission conditions and H.264/AVC video codec used for all presented experiments. *Error detection* aims at utilizing the damaged packets for decoding; it analyzes possible gains and ways how to extract the possibly correct information. *Error concealment* provides an extensive experiment-based review of well-known error concealment methods combined with novel solutions and techniques, applicable for real-time wireless transmission systems. *Cross-layer design* proposes improvements for H.264/AVC video transmission over UMTS, based on information sharing between layers.

In the following the organization of this thesis is introduced in detail and the contributions of the author are highlighted. Throughout the document, the publications (co)authored by the author of this thesis are marked by red color.

**Chapter 2** presents novel techniques to make use of the information in damaged packets. In the first part, an overview of VLC coding used in H.264/AVC baseline profile is provided. Further, the syntax of H.264/AVC is discussed and residual redundancy is analyzed together with the possibilities of its exploiting for error detection. Error detection using syntax check have been previously investigated for H.263 [61], having much simpler syntax and VLC. The performance of syntax analysis based error detection for H.264/AVC is evaluated in terms of quality and error detection probability [62]. Some of the remaining errors are then detected in the picture domain using a voting-based artifact detection mechanism adapted to the specific form of the decoding impairments [63].

The reliability of syntax analysis based error detection is limited and desynchronized decoding may still lead to annoying impairments. Thus, the next step is to limit the size of the distorted area resulting from decoding of a damaged packet. A VLC resynchronization mechanism is proposed that signalizes the resynchronization point positions [64] instead of state-of-the-art inserting of synchronization marks [38].

To enable error concealment of resynchronized erroneous video parts, a reliable error detection is necessary. Apart from the parity bit based approach [64], an error detection mechanism using a novel relation based fragile watermarking is presented [65]. This mechanism provides wide scalability with respect to detection probability and distortion, considerably outperforming state-of-the-art mechanisms [66].

A step further in exploiting of residual redundancy is sequential decoding [67], which was improved and combined with impairment detection together with error concealment [68] and with resynchronization information [69] sending.

**Chapter 3** presents a review of the key techniques for error concealment [48],[49]. Some of them are refined correspondingly to the new features provided by H.264/AVC.

The first part of the chapter is dedicated to spatial error concealment mechanism, including a proposed novel directional interpolation [70]. Spatial error concealment together with temporal error concealment is evaluated in various conditions. Since scene changes substantially impact the performance of error concealment mechanisms, a reliable scene change detection mechanism is proposed [71]. In contrary to the popular mechanism utilizing the encoded bitstream for cut detection [117], the proposed mechanism operates in pixel domain, avoiding failures coming from various encoder settings.

An adaptive real-time capable algorithm, which chooses a suitable error concealment method for each macroblock, is finally chosen [72] and evaluated with H.264/AVC. The appropriate decision rules are discussed and tested.

**Chapter 4** presents several improvements resulting from the proposed cross-layer design. Following the results from Chapter 2 a lower layer packet based error detection and a VLC resynchronization mechanism is presented [73]. This mechanism is improved by using of link error prediction [74] and attaching the side information only if channel conditions are bad [75].

The content awareness is further exploited by a link layer scheduling mechanism prioritizing the more important packets [76],[77] according to the channel conditions. The decisions based on video semantic are further enhanced by employing a novel distortion model [78].

**Chapter 5** summarizes the presented error resilience techniques, error concealment mechanism and network aware improvements. It provides an outlook to their possible deployment, some general conclusions and final remarks.

**Appendix A** lists the nomenclature as well as the abbreviations employed throughout this thesis.

**Appendix B** lists the video sequences utilized in this thesis for testing and evaluation of the proposed methods and provides details of their spatial and temporal characteristics.

**Appendix C** provides an overview of UMTS with focus on the radio access network.

**Appendix D** presents the error models employed in this thesis at the bit stream level, UMTS link level and transport level. The influence of correct error modeling on the quality of video is shortly summarized [79]. The mechanism for calculating the expected failure ratio is explained for the UMTS DCH in static scenario [74].

# Chapter 2

# Error Detection

## Contents

**2.1 Entropy Coding in H.264/AVC . . . . . . . . . . . . . . . . . . . . . . . . . . 22**
  2.1.1 Encoding . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 22
  2.1.2 Consequence of Errors . . . . . . . . . . . . . . . . . . . . . . . . . . . 23
**2.2 Error Detection by Syntax Analysis . . . . . . . . . . . . . . . . . . . . . . 24**
  2.2.1 Syntax Check Rules . . . . . . . . . . . . . . . . . . . . . . . . . . . . 25
    2.2.1.1 I Frame IEs . . . . . . . . . . . . . . . . . . . . . . . . . . . . 25
    2.2.1.2 P Frame IEs . . . . . . . . . . . . . . . . . . . . . . . . . . . . 26
    2.2.1.3 Slice Header IEs . . . . . . . . . . . . . . . . . . . . . . . . . . 26
  2.2.2 Error Handling . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 26
  2.2.3 Performance Evaluation . . . . . . . . . . . . . . . . . . . . . . . . . . 27
**2.3 Error Detection by Detection of Impairments . . . . . . . . . . . . . . . . . 30**
  2.3.1 Detection in I Frames . . . . . . . . . . . . . . . . . . . . . . . . . . . 31
  2.3.2 Detection in P Frames . . . . . . . . . . . . . . . . . . . . . . . . . . . 33
**2.4 Encoder Assisted Error Detection . . . . . . . . . . . . . . . . . . . . . . . 34**
  2.4.1 Error Detection by Checksums . . . . . . . . . . . . . . . . . . . . . . . 34
  2.4.2 Watermarking for Error Detection . . . . . . . . . . . . . . . . . . . . . 35
    2.4.2.1 Force Even Watermarking (FEW) . . . . . . . . . . . . . . . . . . . 36
    2.4.2.2 Relation Based Watermarking . . . . . . . . . . . . . . . . . . . . 39
**2.5 VLC Resynchronization . . . . . . . . . . . . . . . . . . . . . . . . . . . . 43**
  2.5.1 Limitation of Distorted Area . . . . . . . . . . . . . . . . . . . . . . . 44
  2.5.2 Length Indicator Encoding . . . . . . . . . . . . . . . . . . . . . . . . . 45
  2.5.3 Performance Evaluation . . . . . . . . . . . . . . . . . . . . . . . . . . 48
  2.5.4 Watermarking and VLC resynchronization . . . . . . . . . . . . . . . . . . 49
**2.6 Sequential Decoding . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 49**
  2.6.1 Sequential CAVLC Decoder . . . . . . . . . . . . . . . . . . . . . . . . . 50
  2.6.2 Additional Synchronization Points . . . . . . . . . . . . . . . . . . . . . 52
  2.6.3 Postprocessing . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 53
  2.6.4 Performance . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 53

TYPICALLY , error detection capability is provided by a checksum. If an error is detected within a packet, the entire packet is discarded although it may contain useful (non-degraded) information. A checksum can be inserted into the data flow at different protocol layers. For packet video transmission, UDP protocol provides a checksum. The content of the UDP packets determines thus the size of the lost area in the video sequence. Reducing the size of UDP packets limits decreases the missing picture area, but on the other hand also increases the necessary rate due to the protocol headers. This chapter is dedicated to alternative error detection mechanisms that exploit the residual and additional redundancy in the received video stream to detect the errors inside the damaged video packets. the First, a short overview of the entropy coding mechanism in the H.264/AVC baseline profile is given. The syntax of the entropy mechanism is then utilized to localize errors within a damaged packet. This mechanism is later enhanced by additional localization method based on impairment detection in pixel domain. To limit the size of picture areas as a consequence of a bit error, transmitting of resynchronization information is proposed and analyzed. To improve error localization possibilities within damaged packets, encoder assisted methods are investigated. The chapter closes with sequential decoding mechanism improved by impairment detection and resynchronization.

## 2.1 Entropy Coding in H.264/AVC

THE H.264/AVC standard provides two options for entropy coding: Context-Adaptive Binary Arithmetic Coding (CABAC) and Context-Adaptive Variable-Length Coding (CAVLC). CABAC provides better coding efficiency than CAVLC, whereas CAVLC is less complex. Since CABAC is not supported by all profiles, in this thesis CAVLC is focused on.

### 2.1.1 Encoding

In fact, the H.264/AVC CAVLC option includes two methods of entropy coding. The default entropy coding method uses a single infinite-extend codeword set for all Information Elements (IE), carrying signalization information other than the encoded picture data. Thus, instead of designing different VLC tables for each syntax element, a universal *exponential Golomb code* (called also exp-Golomb) was adopted, having a very simple and regular logical structure consisting of a predetermined code pattern.

Each exp-Golomb codeword embodies the following elements:

$$\underbrace{0_1 \ldots 0_M}_{\text{M zeros}} 1 \quad \underbrace{b_1 \ldots b_M}_{\text{M info field bits}} \quad .$$

The first $M$ zeros and the one in the middle are regarded as a prefix while the following $M$ bits represent the `info` field carrying the information. The first codeword has no leading zero or trailing information; the second and third codewords have a single-bit `info` field; and so on. The length of each codeword is $2M + 1$ bits. Each Exp-Golomb codeword can be constructed at the encoder based on its index $i \in [0, \infty)$ by calculating

$$M = \lfloor \log_2(i + 1) \rfloor \, ,$$

and putting it into the unary representation, which forms the prefix. The suffix is formed as a binary representation of the $M$ least significant bits of $i + 1$. According to the information element being encoded, its value $v$ is mapped on the index $i$ as unsigned ($i = v$) or signed ($i = 2|v|$ for $v > 0$, $i = 2|v| - 1$ for $v < 0$, and $i = v$ for $v = 0$). The third possibility is predefined mapping of an information element value onto a table index, other than unsigned or signed. Each of these mappings is then designed to produce short codewords for frequently occurring values and longer codewords for less common parameter values.

To encode the transformed residuals, a context-adaptive VLC is used. In this technique, VLC tables for various syntax elements are switched according to the values of the already transmitted syntax elements. Since the VLC tables are well-designed to match the corresponding conditioned statistics, the entropy coding performance is improved in comparison to schemes using a single VLC table.

First, a VLC encodes the number of nonzero coefficients and the number of trailing ones (±1) (up to three coefficients; if there are more than three trailing ±1s, only the last three are treated and any others are coded as normal coefficients). There are four possible look-up tables (for encoding of the total number of nonzero coefficients and number of trailing ones), the choice of which depends on the number of nonzero coefficients in upper and left previously coded submacroblocks (context adaptive). Three of the possible tables are variable length code tables, one belongs to a fixed-length code (FLC)[1]. The signs of the trailing ones are then encoded in reverse order, starting with the highest-frequency, using one bit per coefficient. After that, the values of the remaining nonzero coefficients (called *levels*) are encoded, starting with the last coefficient preceding the trailing ones, corresponding to higher frequencies. The choice of the one of seven VLC tables to encode each level adapts on the magnitude of each successive coded level (thus context adaptively again). As a next step, the total number of zeros before the last coefficient is encoded with a VLC. The reason for sending separately the total number of zeros (even if each run of zeros will be encoded as well) is that many blocks contain a number of nonzero coefficients at the start of the SMB and this approach means that the corresponding zero-runs with length zero need not be encoded. Subsequently, the number of zeros preceding each non-zero coefficient (called *run*) is encoded in reverse order using VLC depending on the number of zeros that have not yet been encoded and the value of the run.

### 2.1.2 Consequence of Errors

A bit inversion within an entropy encoded NALU can have different consequences depending on the position where it occurs. The "where" at the VLC reading level refers to the part of the codeword, and at the decoding level to the type of the information element. As an example, let `000010111` be an exp-Golomb codeword. If an error occurs in one of the first four bits, the length of the codeword will be determined wrongly, as well as the begin and length of the next codeword, etc., thus the error will desynchronize the whole decoding process. With an error in the third bit the example codeword will become `001010111` and it will be segmented in three exp-Golomb codewords `00101`, `011` and `1`, corresponding to three values of three information elements. On the other hand, if an error occurs in

---

[1] An FLC is used for the number of nonzero coefficients greater than seven, the three VLC tables for the lower values ({0,1},{2,3},{4,5,6,7} for VLC0, VLC1 and VLC2, respectively).

the last four bits of the example exp-Golomb codeword, the value of the decoded IE will be wrong, but it will have no influence on the reading of the following codewords. However, at the decoding level, the wrong value of the IE can also have consequences for the interpretation of the following IEs.

The consequences of errors in a CAVLC encoded NALU can be subdivided into the following four categories:

1. **Illegal codeword**: The sequence of bits to be read does not correspond to any possible codeword. The rest of the NALU cannot be unambiguously decoded because the start of the next codeword is not known.
2. **Length mismatch**: The length of the NALU does not correspond to the length of the slice.
    - All macroblocks were decoded, but there still remains a part of NALU to be read.
    - The reading of NALU was finished, but there still remain macroblocks to be decoded.
3. **Decoding failure**: The NALU can be read, but the decoder cannot decode it, because the parameter value or a combination of parameter values is not allowed.
4. **Undetectable error:** The NALU can be read and the decoder can decode it although an error occurred.

Clearly, the first three categories mentioned above provide a potential to detect at least the presence of an error.

## 2.2 Error Detection by Syntax Analysis

IN the previous section, the consequences caused by an error in an entropy encoded NALU were subdivided into four categories. Category 1 serves naturally as an error detection mechanism — if there is an illegal codeword, obviously, an error must have occurred either in the codeword being decoded or before. The same applies for Category 2 and 3. Category 4 still leaves some space for further analysis: The values of IEs can still be checked from the contextual point of view (e.g. feasibility of the IE combinations, or consistence with the signalized profile/level). Application of all these rules is called *syntax analysis* in this work.

Syntax analysis alone does not provide means to exactly identify the position of the error. Thus, it is not possible to systematically correct the errors at the bit level (an extended approach is the sequential decoding of VLC discussed in Section 2.6). In some cases an error can be detected within the codeword where it happened. In the majority of cases, however, the consequences of the error will be detected later or not at all. The decoding of a slice containing an error at the position $b$ can be described by three intervals illustrated in **Figure 2.1**:

- **[a,b):** The slice is correctly decoded from its begin $a$ up to the error at the position $b$.
- **[b,c):** The error remains undetected until the position $c \geq b$, this part is decoded incorrectly.
- **[c,d]:** The part starting with position $c$ until the end of the slice $d$ cannot be decoded correctly, thus an error handling routine is called.

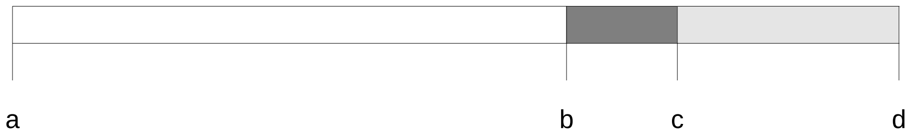


**Figure 2.1**: Intervals corresponding to correctly decoded, incorrectly decoded and concealed parts of a slice.

The size of the interval $[b, c)$ determines the performance of the syntax check. The smaller the interval, the lower will be the amount of undetected and thus unconcealed remaining artifacts. The size of the interval $[c, d]$ influences the efficiency of error concealment — the smaller the interval, the more accurate will be the reconstruction resulting in higher quality. The size of the interval $[c, d]$, however, depends on the position of the error within the packet. For the same position of the error within the packet, the better syntax analysis works, the longer will be $|c - d|$. It is assumed that error concealment performs better than direct decoding of the slice.

### 2.2.1 Syntax Check Rules

Some simple syntax check rules for H.263 were proposed in [61]. However, H.264/AVC has a completely different VLC structure. Therefore, detailed syntax analysis rules were proposed and evaluated in [62]. They are summarized in the following for the common IEs of I frames, P frames, and slice header. In general, the exp-Golomb encoded IEs cannot contain an illegal codeword. An error may, however, cause length mismatch (if the error occurred in prefix), decoding failure, or remain undetectable (if the error occurred in suffix).

#### 2.2.1.1 I Frame IEs

`mb_type`
Detectable if the exp-Golomb encoded value is out of range.

`intra4x4_pred_mode`
The intra 4×4 prediction mode IE is encoded by a codeword of fixed length of one (1) or four ($0b_1b_2b_3$) bits. An error is detectable only by a contextual analysis. Spatial prediction uses reference to the surrounding macroblocks. If they are not available, not yet decoded or belonging to another slice, a contextual error is detected.

`intra_chroma_pred_mode`, `coded_block_pattern`, `mb_qp_delta`
Detectable if the exp-Golomb encoded value is out of range.

`Luma(Chroma) # c & tr.1s`
The VLC look-up table used for this IE is not complete, illegal codewords are possible.

`Luma(Chroma) trailing ones sign`
The signs of the trailing ones are fixed-length encoded and do not influence any of the following parameters. By means of syntax check it is not possible to detect errors in this IE.

`Luma(Chroma) lev`

Decoded pixels can only take values lying in the range [0,255]. During VLC reading, values outside the bounds are immediately associated to errors. During decoding, the residuals are added to the predicted values and the check is repeated. Here, an extended range $[-\lambda, 255+\lambda]$ is considered due to quantization offset (with $\lambda$ being the value of this IE).

`Luma(Chroma) totalrun`

The VLC look-up table used for this IE is not complete, illegal codewords are possible.

`Luma(Chroma) run`

Depending on the number of remaining zeros, a VLC table is chosen. For more than six remaining zeros a single table covering the zero run range [0,14] is used. Therefore, the out of range errors can be detected as well as illegal codewords.

#### 2.2.1.2 P Frame IEs

`mb_skip_run`

The number of MBs within a frame is known. The number of skipped MBs thus cannot be greater than the difference between the total number of MBs and the number of already decoded MBs.

`sub_mb_type`

Detectable if the exp-Golomb encoded value is out of range.

`ref_idx_l0`

The index of the reference frame cannot point outside the buffer.

`mvd_l0`

Detectable if the motion vector points to an (S)MB lying completely out of the frame.

#### 2.2.1.3 Slice Header IEs

`pic_parameter_set_id`

The PPS index cannot be greater than the number of defined PPSs.

`first_mb_in_slice`, `slice_type`, `frame_num`

Depending on the GOP structure an out of range error can be detected

`pic_order_cnt_lsb`

Detectable in the context of frame type and slices belonging to the same frame.

`slice_qp_delta`

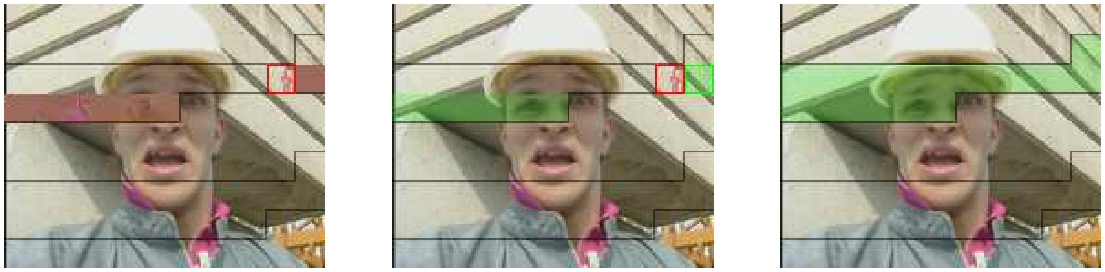
Detectable if the exp-Golomb encoded value is out of range.

### 2.2.2 Error Handling

After detecting an error, the rest of the slice can be concealed or further decoded. It is difficult to guess which of the two error handling strategies will provide better results. Decoding of the possibly desynchronized stream out of a damaged packet may result in impairments that are rather annoying from the point of view of the quality experienced by users. Thus, in most cases an error concealment will improve the performance. **Figures 2.2(a), (b)** and **(c)** illustrate the effect of different error handling on the quality of the picture. The error was inserted into the MB number 24 (marked by the red square), but detected first in the MB number 25 (marked by the green square). Slices are separated by means of black lines. **Figure 2.2(a)** depicts the consequences of the decoding of the entire damaged packet. Such decoding is not possible in all cases — some errors even cause a crash of the decoder. To simulate desynchronized decoding, the decoder has to be modified to support decoding of a damaged packet. The parameter value that would cause crashing of the decoder has to be replaced with the most similar legal one (e.g. a lowest/highest allowed value, a value supported in this level/profile, a value that most probably does not cause a conflict). If the error causes that the whole NALU was read, but there still remain some MBs to be decoded, an error concealment method is called for the missing MBs. If there is, in the contrary, still a part of the NALU remaining although all MBs were already decoded, the rest is simply discarded. **Figure 2.2(b)** shows an example of a decoded frame using such syntax analysis. The macroblock where an error was detected together with all macroblocks remaining until the end of slice is concealed with a simple error concealment, replacing each corrupted macroblock with the spatially corresponding one in the previous frame. **Figure 2.2(c)** illustrates the concealment of the whole slice if an error occurred. In general, such approach works well for sequences with low amounts of movement, where the error concealment methods are able to reconstruct the lost parts almost perfectly. If movement is present, error concealment does not work well and the errors propagating from frame to frame may cause a considerable distortion of the video; the distorted area corresponds to the size of a slice. The decoding of the entire slice out of a damaged packet reduces the erroneous/concealed area although the error amplitude may be higher. However, in a sequence containing more movement, the refreshing of information is faster and thus, distorted area size will decay faster. This can lead to results better than error concealment of the whole slice.

### 2.2.3 Performance Evaluation

To test the performance of the three discussed error handling strategies, the JM decoder was modified. As a test sequence, the "foreman" in QCIF resolution was chosen, encoded using baseline profile. The GOP size was 10 frames. This rather small GOP size was selected in order to test properly the performance of the syntax analysis not only for P frames but also for I frames. The slicing mode with a maximum number of bytes per slice limited to 700 was chosen and the number of reference frames for motion estimation was limited to five. No RDO and no rate control were used. To generate errors, a



(a) Decoding of the entire damaged slice

(b) Decoding up to detected error, then error concealment

(c) Concealment of the entire slice

**Figure 2.2**: Decoding of a corrupted slice using different error handling strategies.

binary symmetrical channel (BSC) (cf. Appendix D.1.1) was employed. Bit error rates (BER) of $10^{-2}$, $10^{-3}$, $10^{-4}$, $10^{-5}$, $10^{-6}$, and $10^{-7}$ were tested. The number of error pattern realizations (decodings) was 75; the results were obtained by averaging over the frame Y-PSNR of all these decoded sequences.

**Figure 2.3** shows the distribution of the $[b, c)$ size in MBs for errors that have been detected in both I and P frames. It represents the delay between the true occurrence of an error and its detection. This delay determines the dimensions of the resulting impairments. According to this EPDF (Empirical Probability Density Function), an error is rarely detected in the same MB in which it occurs. In more than 75% of the cases the $[b, c)$ interval is between 1 and 5 macroblocks long. Thus, the quality of the decoded stream could be improved by an automatic error concealment of one or two macroblocks before the error was detected. For the performance evaluation, however, this improvement was not implemented. Note that with syntax check, no false detections (no error occurred, but it is detected) may happen. Nevertheless, an error may be detected that does not have a visual significance and thus, its error concealment does not necessarily improve the resulting video quality. Such situation is more likely to occur in P frames.

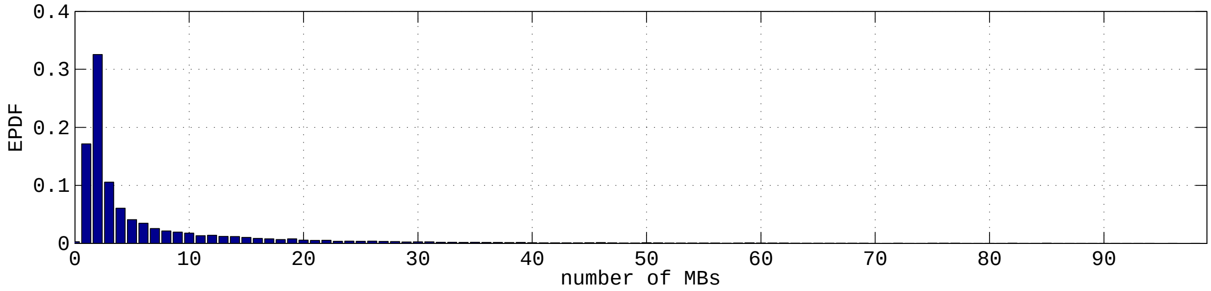
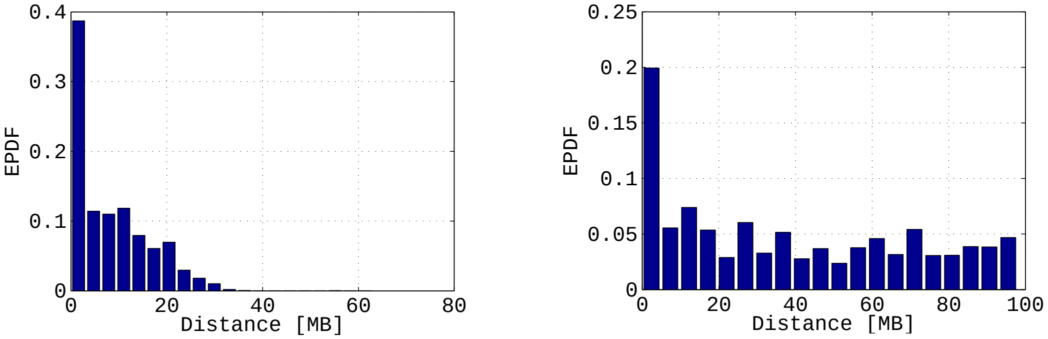


**Figure 2.3**: Detected errors: distribution of the number of macroblocks between the error occurrence and its detection for both I and P frames.

**Figure 2.4(a)** and **(b)** provides information about $[b, c)$ size for undetected errors. It represents the distribution of the size of intervals between the true error occurrence and the end of the slice for I and P frames separately. Note that most of the errors remain undetected since there are only few MBs left until the end of the slice. Such undetected errors do not severely affect the quality of the decoded video. The difference between the $[b, c)$ interval range for I and P frames occurs mainly due to the generally smaller amount of MBs per slice in I frames.

The improvement in the quality measured by Y-PSNR is viewed in **Figure 2.5 (a)** and **(b)** for QP = 20 and QP = 28, respectively. The improvement of quality is slightly higher for QP = 20 than for QP = 28. This is caused by the lower compression ratio at QP = 20, resulting in more VLC encoded information per macroblock, leading to higher probability of correct detection — lower number of MBs between the real error and its detection. The proposed error detection method in cooperation with error concealment clearly outperforms both the decoding and the concealment of the entire slice. The greatest improvement is achieved for the channels affected by BER between $10^{-3}$ and $10^{-6}$, which recalls the mobile wireless channel characteristics.

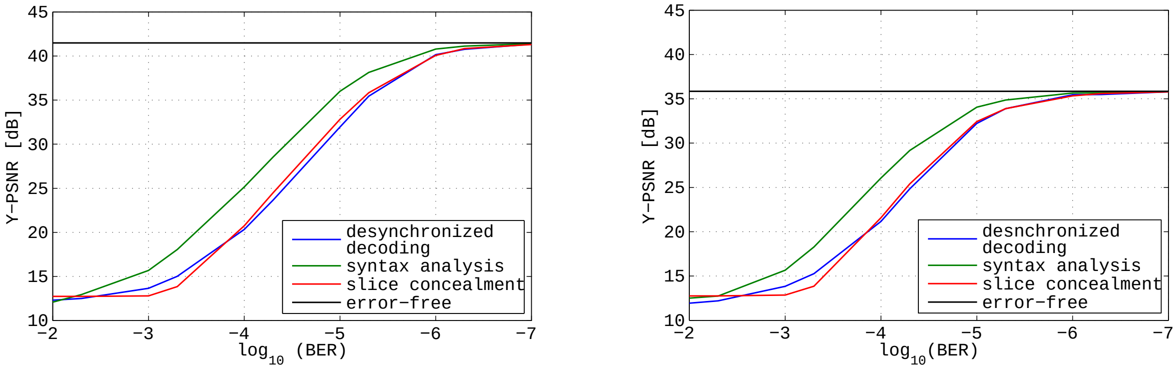
The impact of different error handling strategies on the quality in time is illustrated in **Figure 2.6** for QP = 20 and BER = $10^{-5}$. The effect of the fast motion can be seen between the frames 250 and



(a) I frame

(b) P frame

**Figure 2.4**: Undetected errors: distribution of the interval between the error occurrence and the end of the slice.



(a) QP = 20

(b) QP = 28

**Figure 2.5**: Performance of the different error handling strategies

350 of the “foreman” sequence. Concealment of the entire slice performs in such situations clearly worse than decoding of the entire slice.

In general, syntax analysis based decoding outperforms the common slice concealment strategy. The quality difference between these two methods for the tested configuration is up to 5 dB, a quite significant difference from the point of view of user experienced quality. This is rather surprising, since the undetected errors in the case of syntax check result in artifacts, that are not concealed. The non-concealed artifacts correspond to the interval $[b, c)$ from **Figure 2.1**. Making this interval shorter surely improves the quality. Under the assumption of the same VLC code and stream structure, this can only be achieved by shortening the NALU, or by VLC resynchronization described in Section 2.5. If the artifacts with high magnitude could be detected, error concealment would even more increase the resulting quality. A way of detecting the remaining impairments is discussed in the following section.

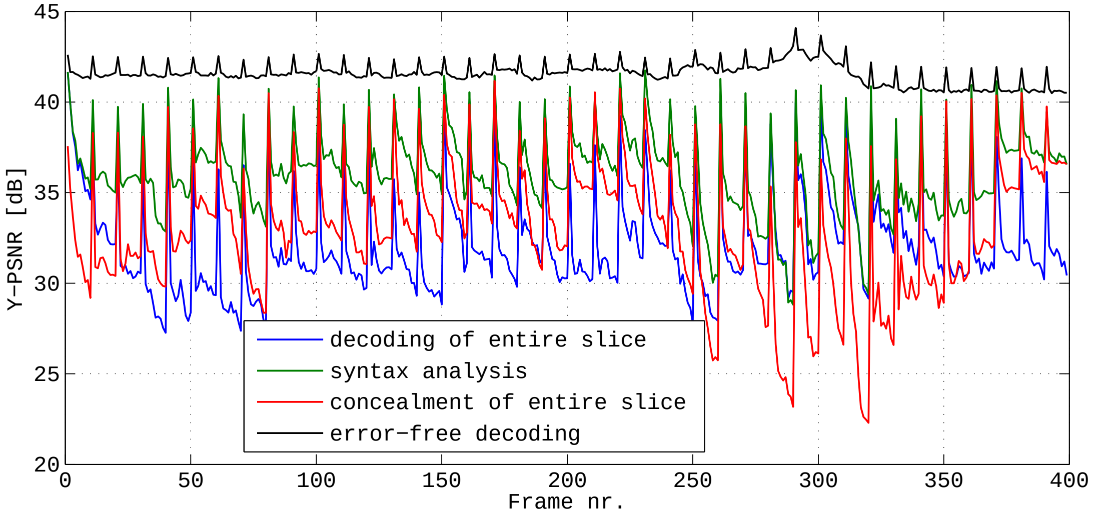


**Figure 2.6**: Y-PSNR over frame number for QP = 20 and BER = $10^{-5}$.

## 2.3 Error Detection by Detection of Impairments

THE impairments caused by video stream decoding out of a damaged packet have some discriminative characteristics that permit to distinguish them from the regular video sequence content.

Typically, the errors in I frames affect larger picture areas than the errors in P frames. Due to both spatial prediction and VLC desynchronization, artifacts in I frames typically propagate until the end of the slice, affecting a considerable part of the frame. In addition, the following frames are also affected, since they refer to the MBs erroneously decoded in the damaged I frames. The temporal error propagation then may involve all following frames until the next refreshment.

The artifacts in P frames seem often to be isolated to a single (S)MB. Since the inter predicted residuals are typically small or even skipped, their errors have often a negligible influence on the quality and on error propagation. The errors in motion vectors and/or reference frame number cause artifacts similar to those of temporal error concealment, i.e. spatial displacement of the affected MBs. The CAVLC desynchronizes rarely.

Some screenshots of the video sequences decoded from damaged NALUs are depicted in **Figure 2.7** for I and P frames and different configurations of the encoder (the first and third picture were obtained with data partitioning — only residuals are affected by error). The first and second picture (from right) illustrate artifacts in I frames, the last two pictures are screenshots of P frames. In the first P-frame screenshot, small impairments around the mouth and chin of the foreman can be seen, caused by errors in partition B/C containing residuals only. In the second P-frame screenshot, there are also artifacts caused by errors affecting motion vectors.

A general detection of artifacts has already been investigated as a means for perceptual video quality estimation. A rather complex, human visual system based method for detection of impairments is proposed e.g. in [9]. The primary goal of the method is detection of the compression artifacts.

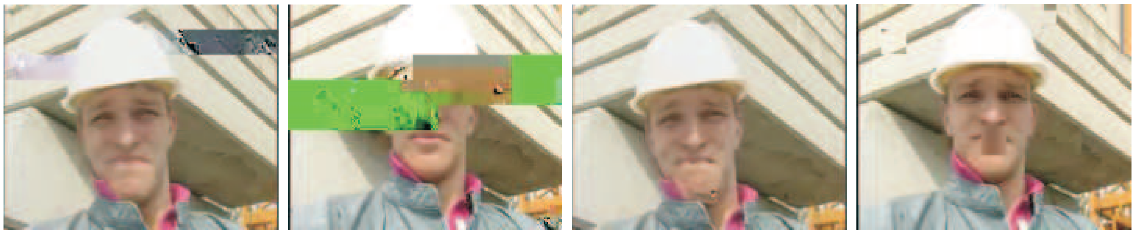


**Figure 2.7**: Screenshots of the video sequences decoded from damaged NALUs.

However, the regular form of the impairments resulting from decoding of damaged packets provides the possibility of locating them via an analysis in pixel (spatial) domain. In [68], such method is proposed for detecting the artifacts remaining after sequential decoding of H.264/AVC CAVLC (cf. Section 2.6). The detection is based on a voting system that analyzes the difference frames and checks the size, shape and propagation of the candidate artifacts. This method is later modified in [63] for detection of artifacts after syntax analysis. Additionally, a novel mechanism is proposed for detection of impairments in P frames. Those methods, operating in pixel domain, are summarized and evaluated in the following.

### 2.3.1 Detection in I Frames

An impairment appearing in I frame differs considerably from the spatially corresponding area in the previous frame. Thus, artifacts can be detected as a kind of anomaly in the picture sequence, using the frame difference. However, great differences may also be observed in the video sequences with high amount of movement, e.g. fast scene changes. Hence, frame difference can indicate an impairment, but frame difference alone is not sufficient for an accurate detection. Fortunately, the affected area is limited by the size of the damaged packet, i.e. limited to the area of one slice. Thus, artifacts in I frames typically have rectangular shape corresponding to several successive blocks affected. This provides more robust means for detection — the areas with high frame difference can be checked to have the right shape and size. Moreover, after receiving the next refreshing information (i.e., I frame), the impairment remaining from the temporal error propagation disappears. This can be used for artifact detection only if the I frame period is sufficiently small so that the sequence does not refresh automatically due to the movement. Refreshing I MBs may also be used similarly, however, in this work they were not considered. The check of artifact disappearing has also the disadvantage of additional delay of the detection, since the decision is performed after the first refresh information received.

Based on the above analysis, first the mean difference between the I frame and the preceding P frame is calculated per block. Then the blocks with difference exceeding the initial threshold $t_{\text{init}}$ undergo a voting based decision process, where the following events contribute to the sum of votes.

- The block-difference exceeds the threshold $K \cdot t_{\text{init}}$, $K > 1$.
- In the block(s), preceding the block in question and belonging to the same slice, an impairment was detected as well.

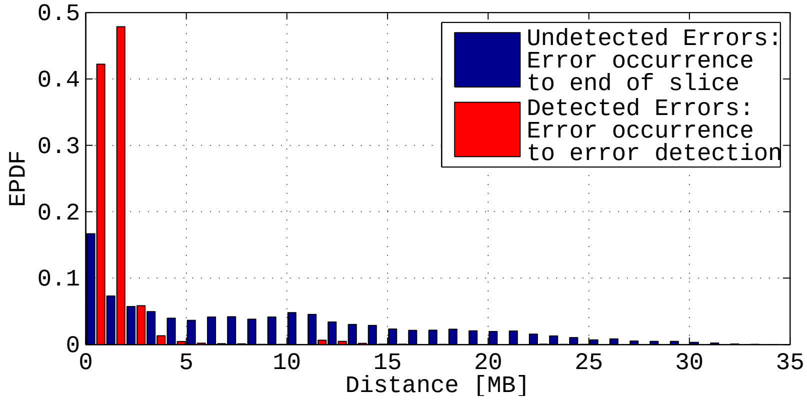


**Figure 2.8**: Performance of syntax check with impairment detection for I frames.

- The impairment disappears with the refreshing information.
- The impairment has a rectangular shape, i.e., it is separated from the rest of the picture by horizontal and/or vertical edges corresponding to block boundaries.
- The impairment does not exceed the spatial boundaries of the slice.
- An a priori information marks the block as erroneous.

The threshold $t_{\mathrm{init}}$ is adapted to the instantaneous amount of motion to ease the correct decisions for sequences with different temporal character. In the presented simulations, the threshold is a 1.5 times the value of the mean difference over past 10 frames. The result of syntax check (detected error) can be considered as an a priori information with the maximum value of vote. If an error is detected by a syntax check, then there is certainly an error. To enhance the reliability of the detection, sequential decoding also uses the a priori information, e.g. the information about the possible FLC bit errors in particular MB from the sequential decoder. For sequential decoding, all above mentioned voting rules were used. The resulting improvement of quality in terms of PSNR is discussed in Section 2.6.

The impairment detection mechanism was tested also in a combination with the results of the syntax analysis. The evaluation was performed over a set of characteristics for sequences of $8 \times 8$ pixels block read in raster scan as described in details in [63]. The implemented voting based system did not make use of the artifacts' disappearing. **Figure 2.8** shows some statistical results obtained for the "foreman" video sequence encoded using QP of 28. As an error model, 40 realizations of BSC with BER of $10^{-5}$ were used. For detected errors, the distribution of the interval size between the occurrence and the detection of an error is shown. The detection delay after impairment detection is much lower than after syntax check alone (cf. **Figure 2.3**). For undetected errors, the distribution of the distance between the error occurrence and the end of the slice is shown. The errors near to the slice end have likely smaller impact on the resulting quality. In contrary to the syntax check, visual impairment detection can also result in a false detections (false alarms). Both tested algorithm implementations (with sequential decoder and with syntax check) resulted in false detection probability of 1–5% depending on the tested error rate. Most of the false detections in the "foreman" sequence were identified (in spite of the motion adaptive threshold) in the frames 270–300, containing fast scene

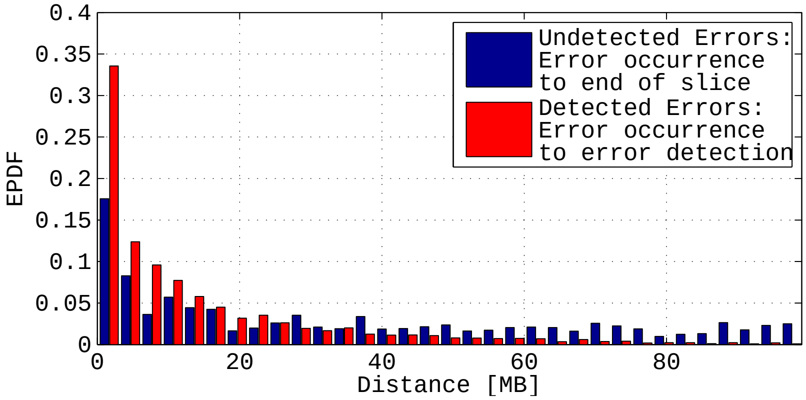


**Figure 2.9**: Performance of syntax check with impairment detection for P frames.

changes.

**Table 2.4** summarizes the results in terms of the probability of detection $P_{\text{det}}$, average detection delay $\Delta_{\text{det}}$, mean distortion in the interval between the error occurrence and detection $\text{MSE}_{\text{det}}$ and mean residual distortion in the interval between undetected errors and the end of the slice $\text{MSE}_{\text{res}}$. Here, the probability of detection $P_{\text{det}}$ denotes the probability of error detection within a slice. The detection delay lays around 1 MB, i.e., the error is typically detected in the same block where it occurs or one block after. The average MSEs for both the detected and undetected errors are much smaller if impairment detection is used. Detection probability rises up to 60% if syntax check is combined with the impairment detection.

| error detection method | $P_{\text{det}}$ | $\Delta_{\text{det}}$ | $\text{MSE}_{\text{det}}$ | $\text{MSE}_{\text{res}}$ |
|---|---|---|---|---|
| syntax analysis | 54.35% | 1.690 | 622.6 | 282.0 |
| impairment detection | 59.99% | 0.925 | 105.3 | 90.3 |

**Table 2.4**: Error detection probability and distortion for I frames.

### 2.3.2 Detection in P Frames

The character and the nature of the impairments in P frames differs from the I-frame impairments. Since for P frames the VLC desynchronizes rarely, it is disadvantageous to apply a combination of syntax analysis and impairment detection. Moreover, syntax analysis may find errors that are not visually significant and thus not detected by impairment detection. The impairments in P frames are often caused by an error in the motion vector, leading to blocking artifacts at the (S)MB boundaries. Blockiness can be detected observing horizontal and vertical edges of the considered frame. The errors in residuals are typically very small and not easy to detect, since they are hardly distinguishable from the regular video content. Thus, the detection of impairments in P frames is based on vertical/horizontal edge detection in combination with a difference frame threshold. No a priori information is used, no slice layout information and no artifact disappearing is considered. The detection delay for detected errors and interval size between error occurrence and end of the slice for undetected errors are depicted in **Figure 2.9**.

**Table 2.5** summarizes the results. The probability of false detection is 3–20% depending on the error rate. The more errors are within the GOP, the higher is the probability of false detection. The main reason for it are the correct MBs that use erroneous MBs in previous frame(s) as a reference for motion compensation. The resulting artifact is similar to errors caused by false motion vectors and thus, it is detected. This problem could partly be removed by a memory storing positions of erroneous MBs from previous frame(s).

| error detection method | $P_{\text{det}}$ | $\Delta_{\text{det}}$ | $\text{MSE}_{\text{det}}$ | $\text{MSE}_{\text{res}}$ |
|---|---|---|---|---|
| syntax analysis | 48.75% | 15.091 | 279.5 | 162.6 |
| impairment detection | 48.54% | 13.741 | 181.9 | 126.7 |

**Table 2.5**: Error detection probability and distortion for P frames.

Despite the slightly smaller detection probability for P frames, $\text{MSE}_{\text{res}}$ for impairment detection is significantly smaller than for syntax analysis.

Both, the syntax analysis and the impairment detection can improve the quality at the decoder. However, their performance decreases with increasing error rate. An efficient method for reduction of the impairment size is to reduce the packet size and thus, to limit the VLC error propagation. This possibility is discussed in the following section.

## 2.4 Encoder Assisted Error Detection

THE error detection methods introduced in Sections 2.2 and 2.3 utilize the correlation of the bitstream and picture domain, respectively. Their performance is rather limited although their employment can be very beneficial in the discussed range of bit error probabilities. To improve the probability of correct error detection and to reduce the detection delay it is proposed to include some redundancy information at the encoder. This can be performed either in the bitstream domain or in the picture (pixel or transform) domain as described in this section.

### 2.4.1 Error Detection by Checksums

In [64], a method based on adding of $k$ parity bits over groups of $M$ VLC encoded macroblocks was proposed to detect errors. The advantage of such scheme is the detection within uniformly sized picture segments, rather than uniformly sized parts of a bitstream. The rate increase caused by adding 1 bit of parity check to each non-skipped macroblock for the "carphone" video sequence in QVGA resolution is shown for different QP values in **Figure 2.10**. Adding more parity bits results in multiples of the rate increase presented for adding one parity check bit.

A single parity bit is capable of detecting any odd number of errors — bit inversions — within a $b$ bits large macroblock. If $P_b$ is the bit error probability (BEP) of the BSC, then the probability of an error within a macroblock is given by $P_{\text{MB}}^{(1)} = 1 - (1 - P_b)^b$. The probability of exactly $l$ errors within a macroblock is $P_{\text{MB}}^{(l)} = \binom{b}{l} P_b^l (1 - P_b)^{b-l}$. Therefore, the probability $P_{\text{res}}$ of non-detected errors

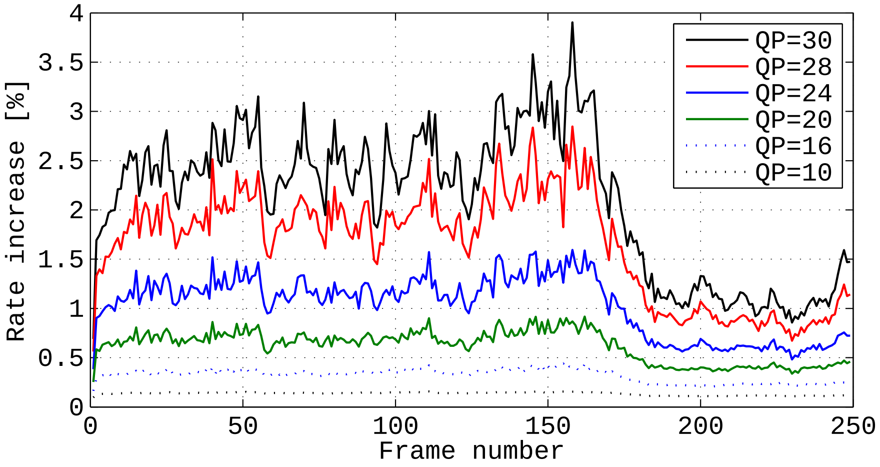


**Figure 2.10**: Rate increase caused by adding 1 bit of parity check to each non-skipped macroblock (”carphone” video sequence in QVGA resolution).

within a macroblock with a single parity bit is given by sum of all probabilities $P_{\mathrm{MB}}^{(l)}$ with an even $l$:

$$P_{\mathrm{res}} = \sum_{k=1}^{\lfloor b/2 \rfloor} \binom{b}{2 \cdot k} P_b^{2 \cdot k} (1 - P_b)^{b - 2 \cdot k}. \tag{2.1}$$

Note that the BEP in this case is not the channel BEP but rather the BEP caused by the VLC desynchronization. Thus the probability of the residual error $P_{\mathrm{res}}$ within the considered macroblock for a single bit parity check lies around 50% (49% for the ”carphone” sequence encoded with $\mathrm{QP} = 28$ and $P_b = 1 \cdot 10^{-6}$, $\mathrm{E}\{b\} = 68$ bits). The probability of detection depends also on the number of bits per macroblock (and thus, on QP), position and number of errors per MB, and on the number of parity bits/checksum. Unfortunately, the loss of macroblocks containing more bits is typically more critical to the resulting distortion, and at the same time an error is more difficult to detect there. Therefore, an alternative mechanism is proposed in the following.

### 2.4.2 Watermarking for Error Detection

An alternative to the previously described bitstream domain redundancy is *watermarking*. Watermarking[2] (WM) is a technique which allows for adding a hidden verification messages to (in this context) video images. Such message is a group of bits that is known and can be verified at the decoder. Watermarking is typically used for authentication and security applications. Watermarking can be *robust* or *fragile* against attacks — changes of image/video content. Furthermore, there is *visible* and *invisible* watermarking, referring to the visibility of the watermark for the human eye.

Watermarking can also be advantageously used for error detection. A WM is inserted in each picture region (i.e., $M$ (S)MBs) the presence of an error should be detected in. After transmission over an error-prone channel, the watermark is checked at the receiver; if it is distorted, an error must have occurred. However, inserting of WM deteriorates the quality already at the transmitter even if

---

[2] The technique takes its name from watermarking of paper or money as a security measure.

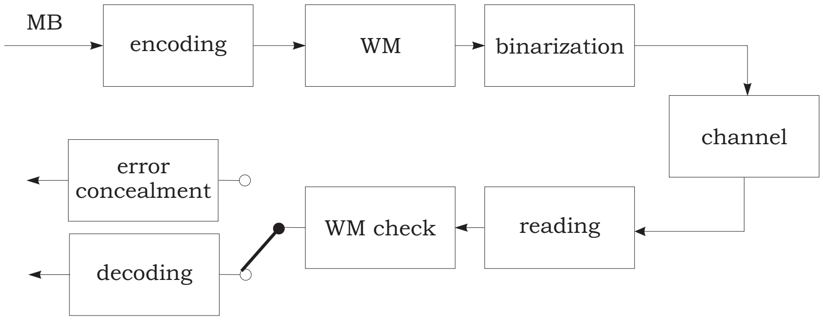


**Figure 2.11**: Functional scheme of fragile watermarking for error detection.

invisible WMs are used. This can be understood as an implicit rate increase: in order to obtain the quality of a watermarked image equal to the quality of the same non-watermarked image, the rate has to be increased. It is assumed that after decoding, the quality of the watermarked video will be higher. Clearly, such method is only beneficial for channels with an error rate higher than a certain threshold.

Watermarking is usually applied to the image either before or after transformation and quantization. Inserting of WMs before transformation and quantization would require a semi-fragile watermarking, robust against transformation and quantization but fragile against errors. Such watermarking at MB level is difficult to design [81]. Insufficient robustness would affect especially videos encoded with higher QPs, and generally, would lead to false detections. Therefore, fragile watermarking inserted after the transformation is considered as more suitable for the purpose of error detection. **Figure 2.11** depicts a generic block scheme of such error detection mechanism. After transformation and quantization, each macroblock (or an $M$-MB region) is watermarked, binarized (entropy encoded and encapsulated in packets) and transmitted over a channel. After reading of the stream, the watermark is extracted and checked. If an error occurs, error concealment is applied. If not, the MB is decoded in a usual way.

The choice of the watermarking method determines the probability of detection as well as rate increase as discussed in the following.

#### 2.4.2.1 Force Even Watermarking (FEW)

The authors of [66] were the first to propose watermarking as a means for error detection. They chose *force even watermarking* and inserted it in each $N \times N$ pixels large H.263 block containing DCT coefficients $a_j, j \in [1, N^2]$ after quantization (the experiments in [66] were performed for $N = 8$). The WM is not necessarily inserted in the entire bandwidth. The DC coefficient is never changed, the AC coefficients are modified from the position $p$ on, in the order of the zig-zag scan. Each $a_i$ with $i = p \dots N^2$ is transformed into a watermarked coefficient $a_i^{(w)}$ according to the following relation

$$a_i^{(w)} = \begin{cases} a_i & ; \quad |a_i| \bmod 2 = 0 \\ a_i - \text{sign}(a_i) & ; \quad |a_i| \bmod 2 = 1, \end{cases} \tag{2.2}$$

where mod 2 denotes the modulo 2 operation and

$$\operatorname{sign}(a_i) = \begin{cases} 1 & ; \quad a_i \geq 0 \\ -1 & ; \quad a_i < 0. \end{cases} \tag{2.3}$$

In summary, all odd coefficients are set to their lower even absolute values. Note that the coefficients with value ±1 become zero allowing for higher entropy coding gain. Thus, the rate of the watermarked sequence will typically be lower than the rate of the original.

In [66] it was already shown that in H.263, FEW brings considerable improvement over the simple syntax check mechanism. In H.264/AVC the DCT transform is applied to blocks of size 4 × 4. In order to test FEW for H.264/AVC, it was implemented into the JM and applied to these blocks. The probability of detection using FEW strongly depends on the start position $p$ and on the content of the video sequence, since due to efficient prediction mechanism, most of the coefficients (especially the high frequency ones) in the blocks are typically zero. The PMF (probability mass function) of the coefficient values is shown in **Figure 2.12** for QP = 30. Correspondingly to the results in [82], the PMF follows a zero-mean Laplace distribution $f(x) = \alpha \exp(-2\alpha|x|)$ with parameter $\alpha > 0$ depending on the value of QP. For QP = 30, the value of $\alpha = 0.5$ was obtained by least-squares fitting. As zeros are encoded in a run-length manner, no watermarking can be applied in such case.

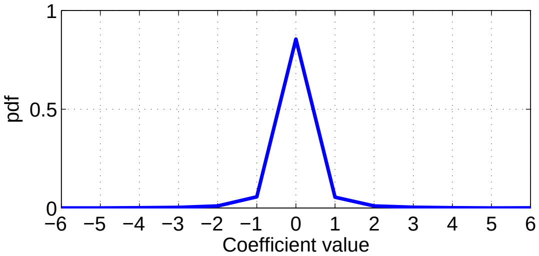


**Figure 2.12**: Distribution of the quantized transformation coefficients values for QP = 30 and the QVGA “carphone” video sequence.

In [66] it was already shown that FEW brings a significant improvement over the simple syntax check mechanism. However, its detection possibilities are limited, too. If an error occurs resulting in an even coefficient, it remains undetected. Moreover, FEW allows to control the distortion only by means of changing the start position $p$. The influence of parameter $p$ on the probability of detection for different QPs is illustrated in **Figure 2.13** for the “carphone” video sequence and BER = $10^{-6}$. Here, the errors detected within the same MB are considered as detected (even if WM was applied to the SMBs). An unpleasant effect of FEW is the decrease of detection probability with an increasing QP. Changes to coefficient values in a picture encoded with higher QP has larger impact on the quality. Syntax analysis had a negligibly small probability of an error detected within the same MB; the corresponding detection probability achieved by impairment detection lays between 30 and 40% for QP = 28. Hence, FEW with $p < 7$ clearly outperforms the impairment detection in the sense of detection probability. Note that similarly to the syntax analysis, there are no false detections possible with fragile watermarking. The price paid for the enhanced detection probability is the distortion at the encoder, that can be equivalently represented as a rate increase necessary to achieve the original

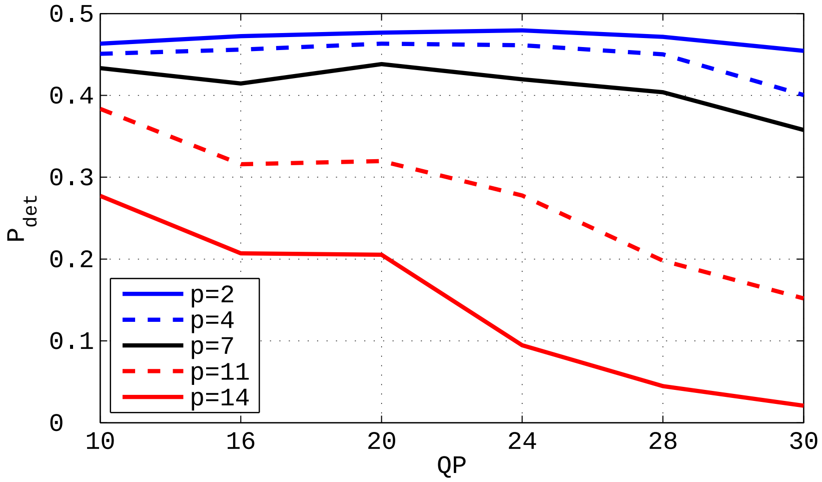


**Figure 2.13**: Probability of error detection for various values of $p$ and QP.

(without WM) quality. **Figure 2.14** depicts the quality degradation of the video at the encoder. The "carphone" video sequence was encoded with QP = 28 and different values of $p$, the WM was inserted in P frames only, there was only one I frame at the beginning of the sequence. These parameters were used in order to allow at least an approximate comparison to the results of [66] achieved using H.263. The degradation at the encoder is about 1 dB of Y-PSNR. Such generic degradation only makes

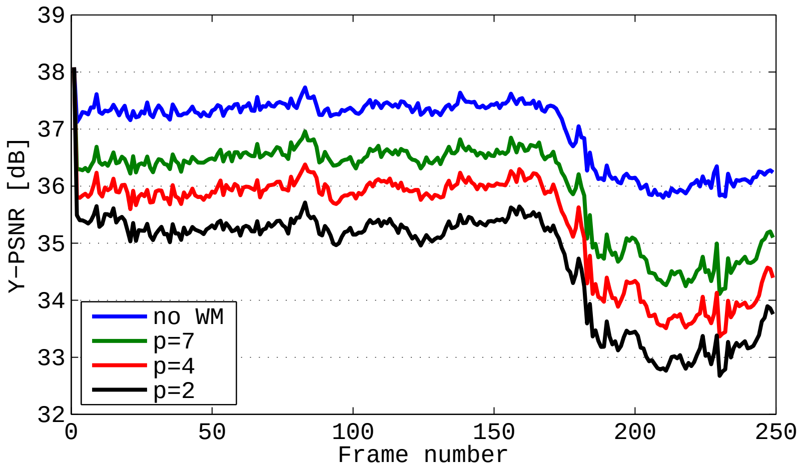


**Figure 2.14**: Y-PSNR per frame at the encoder for "carphone" sequence compressed with QP = 28 and inserted FEW for $p = 2, 4, 7$ and without watermarking.

sense if the quality at the decoder improves. To test the robustness of the investigated methods the transmissions of the video stream over an error-prone environment by a binary symmetrical channel (BSC) with bit error rate of BER = $\{10^{-4}, 10^{-5}, 10^{-6}\}$ was simulated. **Figure 2.15** demonstrates the quality improvement for BER = $1 \cdot 10^{-6}$. The picture areas containing undetected errors that would lead to a decoder crash were in this experiment replaced by a monotone green color. If an error is detected, error concealment is applied from that position until the end of the slice. The results demonstrate a considerable increase of the quality even if the absolute quality is low. The results (relative gains) are similar to the results of [66] for H.263. Whereas the usage of higher values of $p$ is feasible for lower QPs, the typical video streaming scenarios (QP in the interval of 26–30) require $p < 7$. Finally, **Table 2.6** illustrates the rate reduction caused by FEW. There is also a possibility to define an additional stop position, not handled here, since the later coefficients are with high probability zero especially for higher QPs. The lower is $p$, the higher is the distortion at the encoder. From the point of view of the error detection probability, however, inserting the watermark in the significant regions (low frequencies) is beneficial. In the following section a novel scalable fragile watermarking approach is proposed to improve the error detection probability and to allow for better control of the distortion.

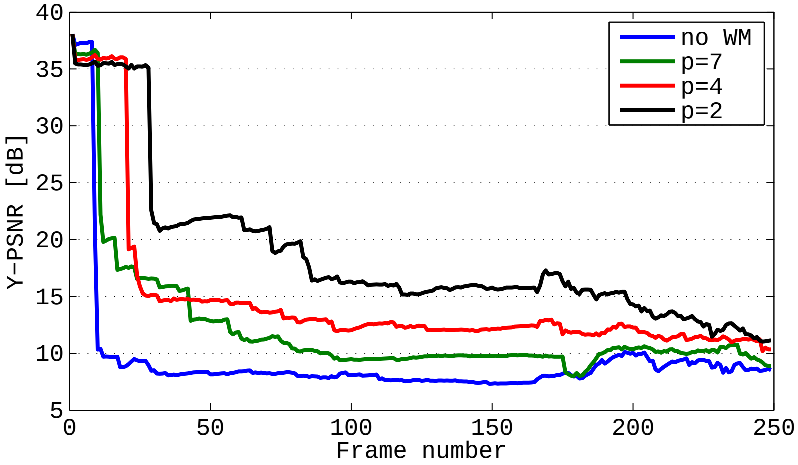


**Figure 2.15**: Y-PSNR per frame at the decoder for "carphone" sequence compressed with QP $= 28$ and inserted FEW for $p = 2, 4, 7$ and without watermarking. BER $= 1 \cdot 10^{-6}$.

| QP | $p = 2$ | $p = 4$ | $p = 7$ | $p = 11$ | $p = 14$ |
|---|---|---|---|---|---|
| 10 | 30.13 | 27.26 | 22.08 | 14.40 | 7.49 |
| 16 | 42.45 | 37.32 | 20.82 | 16.99 | 8.01 |
| 20 | 65.92 | 31.78 | 20.82 | 11.40 | 4.85 |
| 24 | 30.75 | 24.54 | 17.17 | 9.02 | 3.49 |
| 28 | 31.97 | 24.10 | 16.41 | 7.69 | 2.73 |
| 30 | 31.48 | 23.15 | 15.05 | 6.94 | 2.01 |

**Table 2.6**: Average bit rate reduction in % after applying FEW to the "carphone" video.

#### 2.4.2.2 Relation Based Watermarking

The main problem with FEW is its low scalability. Setting all odd coefficients to even has an effect similar to the effect of higher QP. This problem is solved in [65] by introducing the *relation based watermarking* (RBW). This fragile watermarking method in general modifies $n$ chosen DCT coefficients

$\underline{c} = (c_1, c_2, \ldots, c_n)$ to $\underline{\tilde{c}} = (\tilde{c}_1, \tilde{c}_2, \ldots, \tilde{c}_n)$ so that $\underline{\tilde{c}}$ fulfills the predefined equation of the extracting function $f(.)$:

$$f(c_0, \tilde{c}_1, \tilde{c}_2, \ldots, \tilde{c}_n) = K, \tag{2.4}$$

where $K \in \mathbb{R}$ is a constant known to both encoder and decoder. Note that $c_0$ is the DC component that is used in the extracting function. It remains unchanged to avoid the high distortion its change may cause. It is assumed that (2.4) has more than one solutions. From the possible solutions for $\underline{\tilde{c}}$ the one that minimizes the distortion is chosen

$$\underline{\tilde{c}} = \arg\min_{\underline{\tilde{c}}} \|\text{IDCT}(\underline{\tilde{c}}) - \text{IDCT}(\underline{c})\|, \tag{2.5}$$

where IDCT denotes the operation of re-scaling and inverse DCT. Selection of the extracting function $f(.)$ determines the resulting distortion and detection probability. To minimize the distortion a function is required that provides many solutions near to the values of the original coefficients. The number $n$ of the coefficients that can be modified to match the desired extracting function influences the minimum distortion as well. The more coefficients can be modified, the higher is the chance to achieve it with lower distortion. On the other hand, higher $n$ also means higher computational complexity. The $n$ coefficients might be chosen as an arbitrary subset of the (S)MB coefficients. Nevertheless, it makes sense to choose the low-frequency coefficients, since the probability that they are zero is lower. By modifying the zero-valued coefficients, the entropy coding process changes and possibly results in a higher required rate. The influence of $n$ on the error detection probability is not easy to recognize. Although higher $n$ means more coefficients involved in the calculation — thus higher error detection probability, it means also more ways (2.4) can be fulfilled in spite of an error — thus reduced fragility. Fragility directly influences the probability of error detection. Functions that change their output value by a single error are necessary, being robust against multiple errors. To meet all these criteria a modulo $M$ operation (mod $M$) over the sum of the selected coefficients was chosen:

$$f(c_0, \tilde{c}_1, \tilde{c}_2, \ldots, \tilde{c}_n) = \left( c_0 + \sum_{i=1}^{n} \tilde{c}_i \right) \bmod M = K. \tag{2.6}$$

Increasing of $M$ will improve the fragileness of the scheme against multiple errors but on the other hand it will increase the distortion. The choice of $K$ has no influence on the performance.

Apart from fragility, error detection probability, rate, and distortion, computational complexity has to be taken into account when deciding what $n$ and $M$ to choose for a real application. The main source of complexity in the RBW is the RD-optimized WM embedding. The number of WM embedding possibilities $\mathcal{O}_{n,M}$ is given by

$$\mathcal{O}_{n,M} = \binom{n+M-2}{M-1} = \frac{(n+M-2)!}{(M-1)!(n-1)!}. \tag{2.7}$$

In order to choose an RD-optimized solution, $\mathcal{O}_{n,M}$ decodings of each block are necessary. E.g., for $n = 4$ and $M = 4$, 20 encodings are required. This can be reduced by excluding some possibilities that are likely to bring high distortion (e.g. modifying one coefficient only by a high value). Note that for streaming applications, the complexity of WM embedding is not necessarily critical as far as the decoder remains simple. The decoder using RWB only requires calculating (2.6) and checking if the resulting value equals $K$. The value of $K$ is fixed and does not need to be signalized.

In order to evaluate the influence of the parameters $n$ and $M$ as well as QP and BER on the probability of error detection, several experiments have been conducted. The video sequence "carphone" in QVGA resolution was chosen again to enable a fair comparison with FEW. **Figure 2.16** shows the probability of error detection for BER $= 1 \cdot 10^{-6}$ and various values of $n$, $M$, and QP. The results show a very pleasant feature of RBW — the probability of correct detection increases with increasing QP. This is mainly caused by lower number of possibilities that an error results in fulfilling the embedding equation. The absolute value of the detection probability is also considerably higher than for FEW.

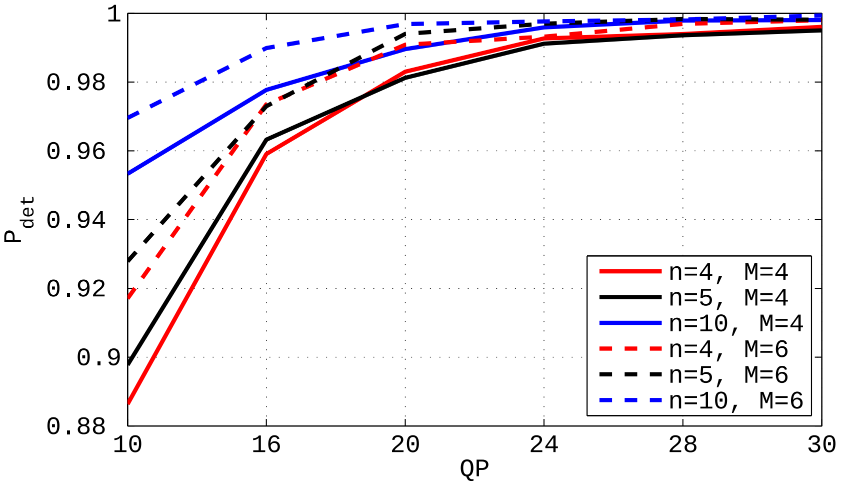


**Figure 2.16**: Probability of error detection for various values of $n$, $M$, and QP.

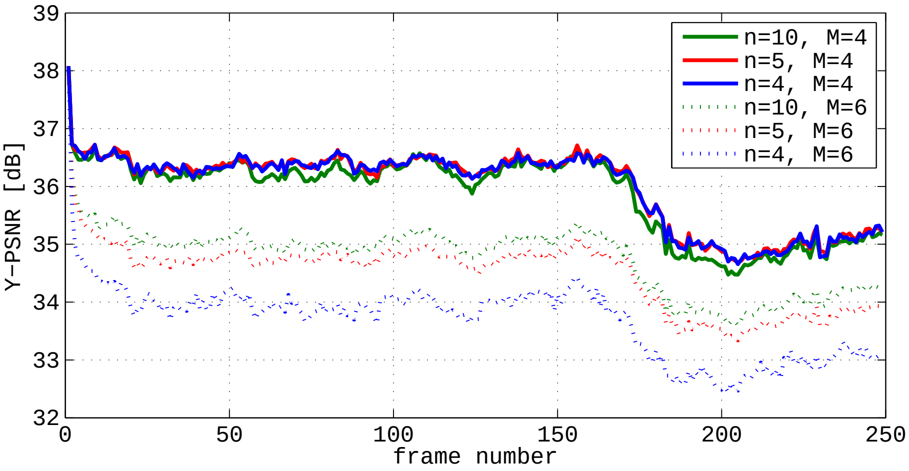


**Figure 2.17**: Y-PSNR per frame at the encoder for "carphone" sequence compressed with QP $= 28$ and inserted RBW for $M = 4, 6$ and different $n$.

**Figure 2.17** presents the Y-PSNR over the frame number of the video sequence at the encoder. The distortion caused by the RBW with $M = 4$ and various values of $n$ corresponds approximately to the distortion caused by FEW with $p = 7$. **Figure 2.18** illustrates the quality at the decoder after a transmission over a channel with BER $= 1 \cdot 10^{-6}$. Following the quite small differences in error detection probabilities as well as different encoder degradation, the resulting Y-PSNR is similar for different values of $n$ and $M$. The combination of $n = 4$ and $M = 4$ seems to perform well enough in spite of the lower complexity of RD-optimized embedding. Note that for all these experiments, the RBW was only embedded in one SMB of the MB. Of course, the embedding with higher $n$ and over more SMBs could provide better results in terms of distortion at the encoder. However, it would be also connected with higher embedding complexity.

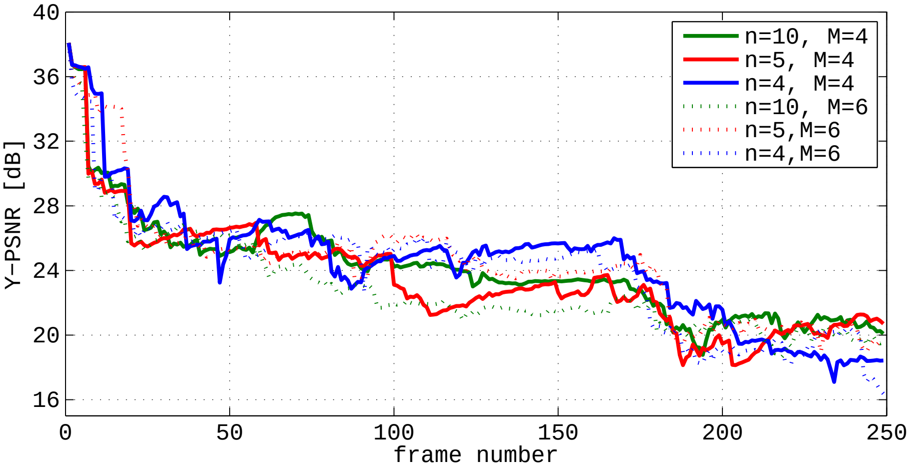


**Figure 2.18**: Y-PSNR per frame at the decoder for "carphone" sequence compressed with QP $= 28$ and inserted RBW for $M = 4, 6$, different $n$. BER $= 1 \cdot 10^{-6}$.

The consequences of RBW embedding on the resulting rate can be seen in **Table 2.7**. As expected, for some cases, especially for higher QPs, the rate increase is considerable. It is caused by forcing different and sometimes higher values to the DCT coefficients. For those cases, macroblock-based checksums may provide a competitive gain. In general, RBW performs considerably better than FEW. To compare fairly and to consider the rate increase caused by using RBW resp. rate decrease using FEW, Figure **2.19** shows the dependency between the size of the compressed and watermarked stream and the Y-PSNR at the decoder for BEP $= 10^{-6}$. Still, RBW brings between 2 and 15 dB

|  | $M = 4$ |  |  | $M = 6$ |  |  |
|---|---|---|---|---|---|---|
| QP | $n = 4$ | $n = 5$ | $n = 10$ | $n = 4$ | $n = 5$ | $n = 10$ |
| 10 | 1.22 | 2.01 | 3.09 | 1.87 | 1.73 | 4.38 |
| 16 | 9.30 | 10.95 | 5.79 | 14.44 | 12.12 | 17.74 |
| 20 | 25.03 | 27.85 | 36.05 | 44.04 | 37.70 | 43.02 |
| 24 | 32.78 | 34.34 | 46.57 | 72.94 | 63.04 | 72.07 |
| 28 | 39.66 | 41.54 | 55.38 | 111.62 | 95.56 | 107.35 |
| 30 | 47.69 | 50.23 | 64.58 | 151.85 | 131.25 | 147.69 |

**Table 2.7**: Average bit rate increase in % after applying RBW to the "carphone" video.

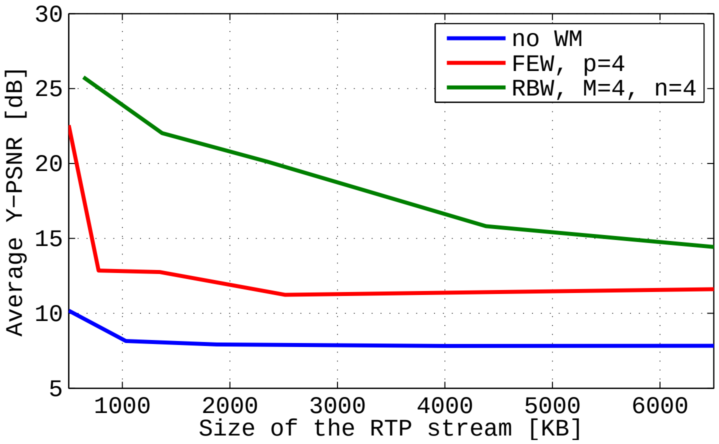


**Figure 2.19**: Comparison of FEW, RBW and packetloss concealment: Y-PSNR at the decoder over the size of the encoded and watermarked RTP stream and BEP $= 1 \cdot 10^{-6}$.

gain against FEW. The gain is slightly higher for higher error probabilities and expected to decrease for decreasing BER. This gain is given by the higher error detection probability of RBW. For example when encoding with QP = 20 we obtained an error detection probability of 98.2% for BEP $= 1 \cdot 10^{-6}$, 98.3% for BEP $= 1 \cdot 10^{-5}$ and 98.6% for BEP $= 1 \cdot 10^{-4}$ with RBW ($M = 4, n = 4$). With FEW ($p = 4$) the probabilities of error detection were about 90%, similar for all three BER values. Single bit parity check only results in about 49% of error detections.

## 2.5 VLC Resynchronization

THE problem of the VLC desynchronization can be solved by different means. In H.264/AVC, the VLC starts at the beginning of each packet. In some literature (e.g. [14]) it is therefore suggested to keep the packet length as low as approximately 100 bytes to achieve robust transmission over error prone channels. The transmission over an RTP/UDP/IPv4 protocol stack is, however, connected with an overhead of 40 bytes per packet (neglecting the slice header, NALU header, and padding; without any other assumption of underlying system[3]). For instance, the 100-byte long packets result in 40% of the rate only for packet headers. The overhead lower than 10% of the data is achieved for packets larger than 400 bytes; the overhead lower than 5% of the data is achieved for packets larger than 800 bytes. Thus, smaller packet sizes are not an efficient solution for the desynchronization problem.

The usage of reversible VLC codes (e.g. [36]) prevents the desynchronization of the decoder. However, the shortcoming of the majority of VLCs with good resynchronization properties is their lower compaction (lossless compression) performance. A fair tradeoff between compression gain and good resynchronization properties represent reversible exp-Golomb [37] adopted in MPEG-4 and H.263+ (Annex D), but not into H.264/AVC where context adaptive VLC schemes provided higher compression gain than a single (R)VLC table.

---

[3] The overhead in context of UMTS is further discussed in [80].

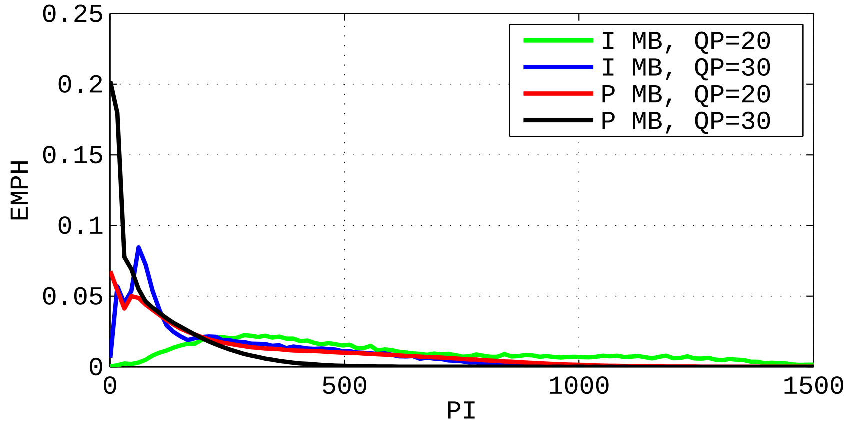


**Figure 2.20**: Distribution of the MB size for intra and inter coded MBs.

Another popular way to increase robustness of source coding schemes is insertion of synchronization marks (SM) [83]. Synchronization marks allow for bit synchronization of the decoder, preventing the error propagation. An RD-optimized mode selection and SM insertion is proposed in [38]. The SMs there correspond to slice header insertion. Apart from the overhead caused by *in-stream* insertion, sending the SMs within the same stream increases the risk of errors resulting in their misinterpretation. In order to avoid these problems, an alternative resynchronization mechanism using *length indicators* (LI) is proposed in this section. The LIs [64] are sent *out-of-stream*, i.e., in dedicated NALUs and require less bits for their encoding.

### 2.5.1 Limitation of Distorted Area

The aim of resynchronization is to limit the size of the distorted picture area which in turn results in better performance of error concealment. In [64] a method of sending position indicator of a VLC codeword start every $M$ MBs was introduced. This method allows for resynchronization of the VLC stream at the position of the first VLC codeword that is entirely included within the $M$ macroblocks. The LIs allow to restart the VLC decoder at known positions and thus confine the propagation of decoding errors which are thereby also easier to detect and conceal.

A tradeoff has to be found between the overhead and the frequency of the LIs determining the size of the erroneous area in (macro)blocks. The overhead $O_{\mathrm{MB}}$ caused by sending a LI of $m$ bits after each $M$ macroblocks is

$$O_{\mathrm{MB}} = \frac{m}{M \cdot \mathrm{E}\{L\}}, \tag{2.8}$$

where $L$ is a random variable representing the size of a macroblock in bits. The distribution of $L$ differs considerably for I and P frames. Furthermore, it varies also for different QPs as well as for different video sequence contents. This is illustrated in **Figure 2.20**, obtained by analysis of the "soccermatch"video sequence encoded by JM in baseline configuration. The mean P MB size $\bar{l}_{\mathrm{P}}$ for the P MBs calculated for the "soccermatch" sequence is 86.24 bits for QP = 30 and 269.37 bits for QP = 20. The mean I MB size $\bar{l}_{\mathrm{I}}$ is 240.30 bits for QP = 30 and 591.75 bits for QP = 20.

To convey the LI information to the decoder without errors, it could be appended to the A packets if DP is used; however, that is not standard-compliant and might break compatibility with other receivers. A better way is to use a reserved NALU type, which will be ignored by receivers not knowing how to handle it. The additional packetization overhead is smaller if the side information can be precomputed to generate larger packets. The decoder needs only buffering capability for one additional packet.

To numerically estimate the overhead, the necessary length $m$ has to be determined. This is discussed in the next section.

### 2.5.2 Length Indicator Encoding

There are two possibilities for representing resynchronization information. It can be represented as an absolute *position* of $M$-MB segments within the packet or as a *length* of the $M$-MB segments (and slice header at the beginning) in bits. The first representation is more robust against errors, as the positions are independently represented; the second representation may lead to lower signalling overhead, depending on the applied binarization method. In this section, the signaling of length will be further discussed.

A trivial way to binarize the LIs is fixed-length coding. The range of values for both position and length based LIs is limited by the maximum size of the packet, i.e., in general the MTU size (maximum transmission unit, typically 1500 bytes) minus length of all packet/slice headers. This corresponds to 14 bits necessary for LI encoding, which is still smaller than an average slice header (approximately 25 bits long). A bit error in such an FLC codeword will lead to an erroneous decoding of that codeword only. A bit error in a VLC codeword may cause desynchronization of the decoding, which in turn results in erroneous decoding of the other codewords after the bit error as well. The compression gain of a VLC depends on the distribution of the source symbols, the source symbols being the values of LI.

**Figure 2.20** shows the Empirical Probability Mass Function (EPMF) $p(l_i)$ of macroblock sizes for I and P macroblocks in bits (for QP = 30 and QP = 20). The EPMFs for QP values between 20 and 30 (used for mobile communication) would lay in-between. This EPMF corresponds to the LI distribution if LI represents the length of an MB and $M = 1$. Distributions for $M > 1$ are more flat, there will be less occurrences of small LIs and more of the larger LIs.

The non-uniform distribution of source symbols may provide the possibility for a lossless compression. Let $i$ be an integer value of LIs and $w_i$ the corresponding codeword. For an optimal Huffman binary prefix-free code [84] and for a source $\mathcal{S}$, the mean codeword length $\bar{l} = \mathrm{E}\{\ell(w_i)\}$ is bounded by

$$H(\mathcal{S}) \leq \bar{l} \leq H(\mathcal{S}) + 1, \tag{2.9}$$

with

$$H(\mathcal{S}) = -\sum_i p(x_i) \log_2 p(x_i) \tag{2.10}$$

being the *entropy* of the source $\mathcal{S}$ and $p(x_i)$ is the probability of the occurrence of the $i$th codeword.

However, designing an optimal Huffman code is not feasible since the LI distribution differs considerably for different types of frame, QPs, and values of $M$ as well as for different sequence contents. If the probability of the source is not known/varying, *codes for integers* [50] are more suitable. In data compression, a code for integers is a prefix code that maps the positive integers $i$ onto binary codewords, with the additional property that the distribution is monotone (i.e., $p(i) \geq p(i+1)$ for all positive $i$). Such code is called universal if for any distribution in a given class, it encodes into an average length that is within some factor of the ideal average length.

There have been several coding schemes for integers proposed so far. The simplest *unary coding* encodes an integer $i$ into a codeword $w_i$ as a sequence of $i$ ones followed by a zero (or equivalently a sequence of $i$ zeros followed by a one). Unary codes correspond to Huffman codes for the semi-infinite alphabet $i \in [0, \infty)$ with probability model $p(i) = 2^{-i}$ and it is thus, for this probability model, optimal. Each codeword $w_i$ has the length $i+1$.

Another wide class of codes combines a unary prefix with other types of code. An example is the *Golomb code* family [85] with an integer parameter $g > 1$. Each integer $i \in [0, \infty)$ is represented by two numbers $q = \lfloor i/g \rfloor$ and $r = i - qg$, where $q$ is represented unary and the reminder $r$ takes the values from $[0, g-1]$. If $g$ is a power of two, $r$ is represented by a $\log_2(g)$ bit binary representation. Correspondingly, each codeword $w_i$ has the length $\lfloor i/g \rfloor + \lceil \log_2(g) \rceil + 1$. If $g$ is not a power of two, still the $\lceil \log_2(g) \rceil$ bits can be used. Alternatively, the required number of bits may be reduced if a $\lfloor \log_2(g) \rfloor$-bit binary representation of $r$ is used for the first $2^{\lceil \log_2(g) \rceil} - g$ values, and the $\lceil \log_2(g) \rceil$-bit binary representation of $r + 2^{\lceil \log_2(g) \rceil} - g$ for the rest of the values. The Golomb code is optimal for the probability model

$$p(i) = (1-q)^{i-1} q, \qquad g = \left\lceil -\frac{1}{\log_2(1-q)} \right\rceil. \tag{2.11}$$

*Elias codes* $\gamma$, $\delta$ and $\omega$ were first proposed in [86]. Elias-$\gamma$ is an equivalent to the exponential Golomb code[4] introduced in Section 2.1. It can be further regarded as a Golomb code with parameter $g$ adapting to $i$ as $g = \lfloor \log_2(i) \rfloor$. The codeword length of an Elias-$\gamma$ encoded integer $i$ is given by $\ell(w_i) = 1 + 2 \cdot \lfloor \log_2(i) \rfloor$.

An Elias-$\delta$ code is also composed of two parts — the Elias-$\gamma$ encoded length of the binary representation of integer $i$ and the binary representation of $i$ without its most significant bit. The codeword length of an Elias-$\gamma$ encoded integer $i$ is given by $\ell(w_i) = 1 + \lfloor \log_2(i) \rfloor + 2 \cdot \lfloor \log_2(1 + \lfloor \log_2(i) \rfloor) \rfloor$.

An Elias-$\omega$ code is composed by more binary represented groups. Each earlier group is the binary encoding of the length less one of the following group. The right utmost symbol is zero, the next group encodes binary the integer $i$, the following group encodes $\lfloor \log_2(i) \rfloor$, etc. The encoding process halts on the left with group of length 2. The length of an Elias-$\omega$ encoded integer $i$ is given by the recursive formula $\ell(w_i) = 1 + \sum_{j=1}^{k} (l_j(i) + 1)$, with $l_1(i) = \lfloor (log_2(i)) \rfloor$, $l_{k+1}(i) = l_1(l_k(i))$, $k \in \mathbb{N}^+$. The summation stops with integer $k$ such that $l_k(i) = 1$.

In **Table 2.8** example codewords of all introduced codes is presented. It can be seen that the codeword lengths $\ell(w_i)$ of the encoded integer $i$ vary for each code. **Figure 2.21** shows the length of codewords for $i \in [1, 100]$ (thereafter the trend of the curves does not change) and all codes presented above. Clearly, the performance of codes in the sense of expected codeword length will strongly depend on the distribution of the source.

The distribution of LI values of I frames for various values of $M$ is not monotone, and it is rather flat. Therefore, for I frames, FLC encoded LIs will probably be the best choice. For P frames, VLC encoding could help reducing the rate.

---

[4] Note that exp-Golomb codes are different from Golomb codes. In literature, these two codes are often confused. E.g., [7] refers incorrectly to [85] as to a source for exponential Golomb codes.

| code | $i = 1$ | $i = 2$ | $i = 3$ | $i = 4$ | $i = 15$ | $i = 32$ |
|---|---|---|---|---|---|---|
| unary | 10 | 110 | 1110 | 11110 | $\underbrace{1\cdots1}_{15}0$ | $\underbrace{1\cdots1}_{32}0$ |
| Golomb(2) | 00 | 01 | 100 | 101 | 111111100 | $\underbrace{1\cdots1}_{15}01$ |
| Golomb(4) | 000 | 001 | 010 | 011 | 111010 | 1111111011 |
| Golomb(8) | 0000 | 0001 | 0010 | 0011 | 10110 | 1110111 |
| Elias-$\gamma$ | 1 | 010 | 011 | 00100 | 0001111 | 00000100000 |
| Elias-$\delta$ | 1 | 0100 | 0101 | 01100 | 00100111 | 0011000000 |
| Elias-$\omega$ | 0 | 100 | 110 | 101000 | 1111110 | 101011000000 |

**Table 2.8**: Example codewords $w_i$ for chosen integers $i = \{1, 2, 3, 4, 15, 32\}$ encoded by unary, Golomb and Elias codes.

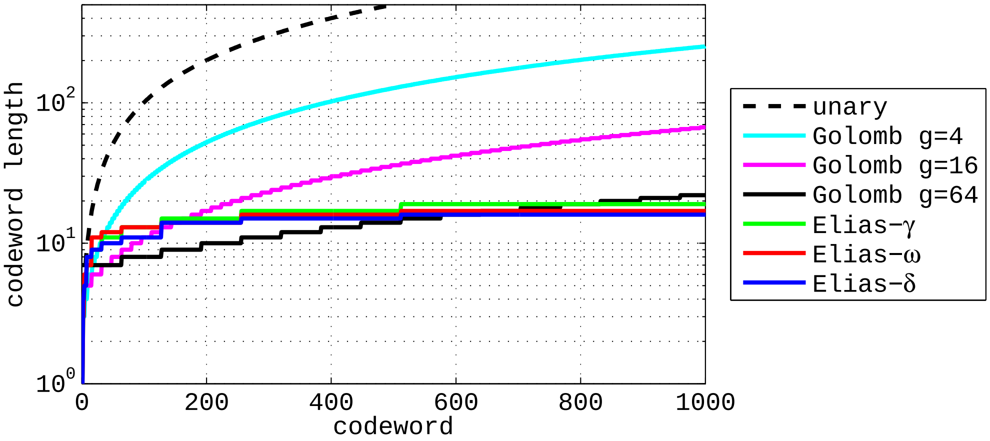


**Figure 2.21**: Codeword lengths for chosen integer codes.

**Table 2.9** contains the expected codeword lengths for all discussed VLC codes applied to the LIs for $M = 1$. The ideal Golomb code is the Golomb code with $g$ resulting in the smallest $\mathrm{E}\{\ell(w_i)\}$. This was different for I and P frames with different QP values: $g_{\mathrm{I20}} = 256$, $g_{\mathrm{P20}} = 127$, $g_{\mathrm{I30}} = 128$, and $g_{\mathrm{P30}} = 62$.

| code | $\bar{l}_{\mathrm{I}}$, QP = 20 | $\bar{l}_{\mathrm{P}}$, QP = 20 | $\bar{l}_{\mathrm{I}}$, QP = 30 | $\bar{l}_{\mathrm{P}}$, QP = 30 |
|---|---|---|---|---|
| unary | 490.22 | 266.29 | 228.90 | 84.59 |
| Golomb (ideal) | 10.45 | 9.61 | 9.34 | 7.83 |
| Elias-$\gamma$ | 16.52 | 13.82 | 13.86 | 9.85 |
| Elias-$\delta$ | 14.33 | 12.29 | 12.43 | 9.20 |
| Elias-$\omega$ | 15.30 | 13.27 | 13.50 | 9.98 |
| entropy | 10.25 | 9.23 | 9.17 | 7.40 |

**Table 2.9**: Calculated average codeword length for chosen integer codes, calculated based on the distribution of LIs estimated from the “soccermatch” video sequence and $M = 1$.

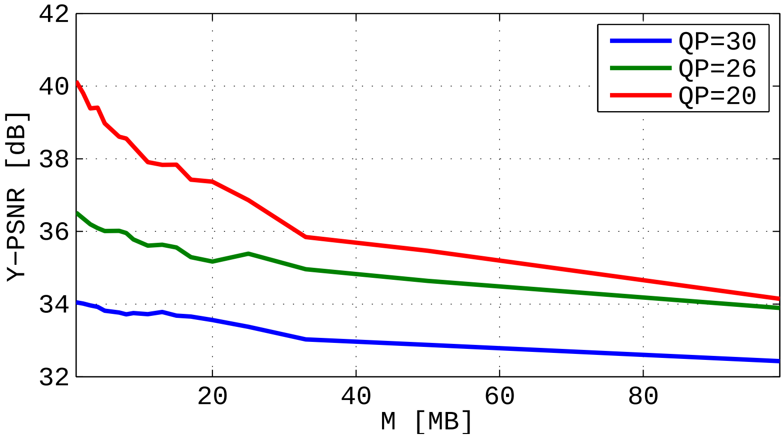


**Figure 2.22**: Y-PSNR after error detection by syntax analysis, depending on the size of the NALU (in MBs).

Golomb codes approach the bound since the distribution of LIs is rather flat. Therefore, the very short codewords are not as beneficial. The codeword set for Golomb codes with high $g$ is similar to an FLC — the length of codewords stays constant over larger ranges of $i$. Thus VLC Golomb coding still benefits over FLC and is clearly worth of using, as it saves for our test cases 3–6 bits per LI.

### 2.5.3 Performance Evaluation

**Figure 2.22** shows the average Y-PSNR if VLC resynchronization is applied together with the syntax analysis. Note that the absolute values of Y-PSNR for $M = 99$ are lower than the values obtained for syntax analysis before. This is because in this experiment, the number of MBs per slice/NALU was limited unlike the size in bytes in previous experiments. The sequence "foreman" was sent over an BSC channel with BER $= 10^{-5}$, for each configuration (maximum number of macroblocks per slice, QP) 20 realizations of the channel were used.

The probability of error detection by syntax check is illustrated in **Figure 2.23**. The probability of detection is small (below 40%) for low values of $M$. However, this decrease is not caused by the lower $M$ itself, but rather by the detection delay of the syntax check as shown in Section 2.2.3. Since the error is detected few macroblocks after its true occurrence, the probability of detection within an $M$ MB long segment is low for lower $M$. This may deteriorate the quality at the decoder, that could be better if an appropriate error detection mechanism were used.

Since both shorter packet sizes and VLC resynchronization require rate increase, **Figure 2.24** compares these techniques normalized by rate.

**Figure 2.24** compares the quality at the decoder resulting from transmitting the VLC resynchronization and the quality resulting from the smaller slice size limit. The comparison is performed normalized on rate, since the two methods result in both, different quality and rate. The different rates follow from different sizes of packets and $M$. For calculating the rate of VLC resynchronization, one packet of LIs per second was taken into account with its complete NALU/RTP/UDP/IP headers. The ideal Golomb code encoding for the LIs was assumed. Definitely, the VLC resynchronization provides better means for limiting the distorted area than smaller packet sizes.

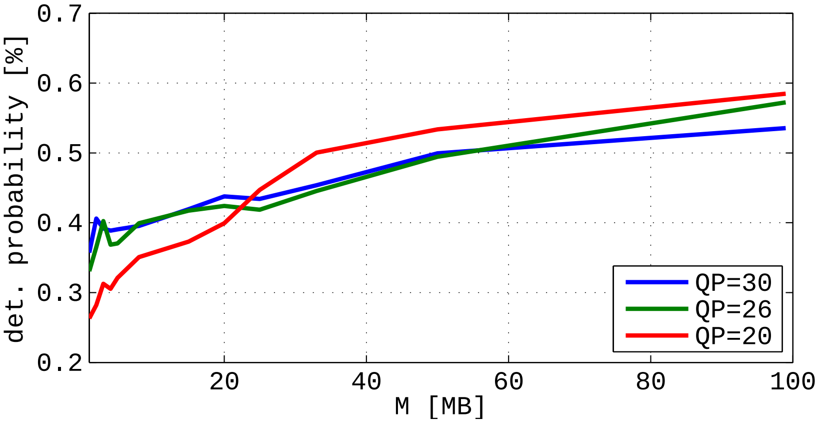


**Figure 2.23**: Probability of error detection within the slice/resynchronized interval using syntax check.

VLC resynchronization may perform even better if combined with a more reliable error detection mechanism with the granularity of the resynchronized intervals. The following section investigates such detection mechanisms as well as their combination with VLC resynchronization. Other applications of VLC resynchronization are discussed in Section 2.6 and Section 4.1.2.

### 2.5.4 Watermarking and VLC resynchronization

Error detection using fragile watermarking only allows to detect errors in residuals, thus it is especially suitable for type B/C packets if data partitioning is used. Of course, it can also be applied without data partitioning. There is even a possibility that errors in IEs other than residuals would be indirectly detected if desynchronization occurred. Desynchronization of VLC leads to annoying impairments as already discussed in Section 2.3. Thus, if an error is detected in an (S)MB, the following (S)MBs until the end of the slice are concealed.

An alternative is sending of side information necessary for the resynchronization as proposed in Section 2.5. If watermarking is additionally applied, the artifacts caused by the possibly desynchronized decoding can be suppressed as the errors are detected.

Finally, **Figure 2.25** illustrates the performance of both the RBW and FEW with and without VLC resynchronization, respectively. The figure was obtained for BER $= 10^{-4}$, the same video sequence "carphone" and QP of 28. The VLC resynchronization provides a gain of several dBs additionally to the watermarking.

The encoder assisted error detection methods together with VLC resynchronization are suitable only for BERs greater than $10^{-7}$. Otherwise, the rate increase is too high for the achieved quality improvement.

## 2.6 Sequential Decoding

SYNTAX analysis as proposed in Section 2.2 demonstrates that there is still a possibility of detecting the errors in the H.264/AVC CAVLC, given by the constraints on the form of the encoded

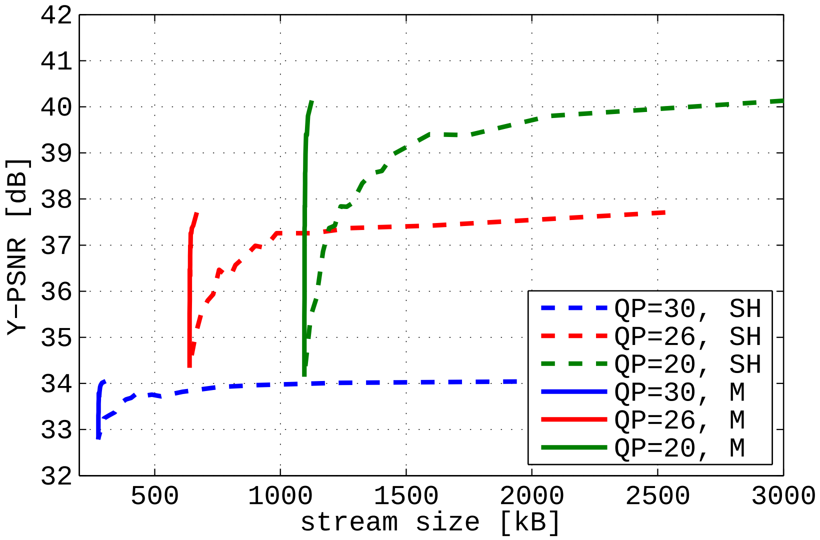


**Figure 2.24**: Quality after syntax analysis detection in relation with the size of the stream for different NALU sizes (SH) and VLC resynchronization (M).

stream that are violated during the decoding. These constraints can also be used to correct errors as demonstrated in this section by means of sequential decoding already presented in [68] and [69]. However, after sequential decoding there still may remain large artifacts if the decoder chooses the wrong path. In this section the focus is given on the improvement of sequential decoding consisting of VLC resynchronization as proposed in Section 2.5 and of impairment detection and concealment as discussed in Section 2.3 since these are the original contributions of the author of this thesis.

### 2.6.1 Sequential CAVLC Decoder

The encoding of residual coefficients depends causally on previous data in the same B/C packet and on the relevant information from A packets if data partitioning is used. This causal code structure is well matched to sequential decoding, which was originally proposed for convolutional channel codes and later refined in [87]. A list of partial decoding paths is kept in memory and each is labeled with a metric that allows comparing paths of different length. Since the list size shall be limited, the decoder needs to decide which paths to explore further based on the path metric. Several strategies exist, one of the simplest involves storing the paths in a stack which is sorted according to the metric. The top path (with the highest metric) is replaced by its extensions (a corresponding number of low-metric paths will be dropped from the stack) and the stack is sorted again. These steps are repeated until the top path has the required length and can be output as the decoded path. The choice of metric determines the performance of sequential decoding. In [88] it was shown that the heuristic metric introduced by [87] minimizes the error probability, provided the so-called “random tail assumption” holds.

Consider a message $w$ that is encoded with the binary variable-length codeword $x_{w,1}x_{w,2}\dots x_{w,\ell(w)}$ and transmitted over a binary-input memoryless channel with transition probabilities $P(y|x)$. The received vector $\underline{y}$ is assumed to be longer than the codeword $\underline{x}_w$. Then the random tail assumption

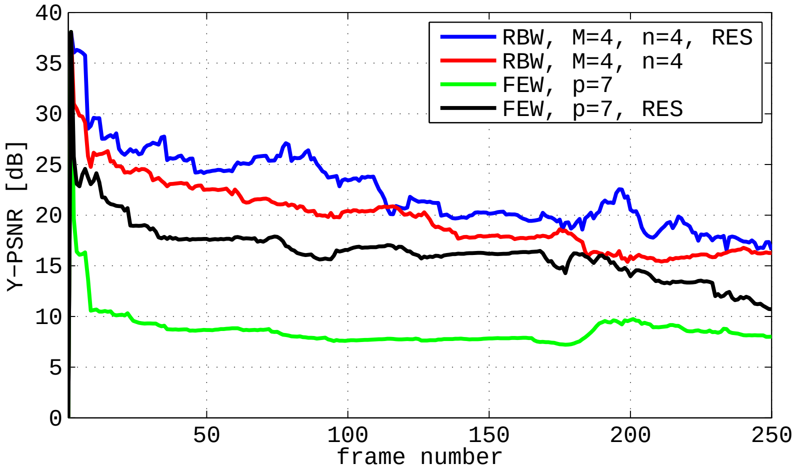


**Figure 2.25**: Quality at the decoder for RBW and FEW, with and without VLC resynchronization (RES), BER $= 10^{-4}$.

states that the bits following the codeword (and belonging to the next codeword) are chosen IID with some probability mass function $Q$. For a good binary source code this is approximately satisfied with equal probabilities of 1/2 for zero and one. Then the *a posteriori* probability that message $w$ has been sent is

$$\Pr(w|\underline{y}) = P(w) \prod_{i=1}^{\ell(w)} \frac{P(y_i|x_{w,i})}{P_0(y_i)}, \tag{2.12}$$

where $P(w)$ is the *a priori* probability that $w$ has been sent and $P_0(y_i) = \sum_x P(y_i|x)Q(x)$ is the marginal channel output distribution induced by $Q$. The metric is now simply the logarithm (usually of base two) of $\Pr(w|\underline{y})$:

$$L(w, \underline{y}) = \log P(w) + \sum_{i=1}^{\ell(w)} \log \frac{P(y_i|x_{w,i})}{P_0(y_i)}. \tag{2.13}$$

Using $Q = (1/2, 1/2)$, the argument of the right-hand "channel term" can be directly computed from the soft inputs, e.g. the LLRs $\log \frac{P(y_i|1)}{P(y_i|0)}$. Extending the metric to sequences $w_1^k = w_1 w_2 \dots w_k$ is straightforward: the *a priori* term $\log \Pr(w_1^k)$ can be decomposed into the sum $\sum_{i=1}^{k} \log \Pr(w_i|w_1^{i-1})$, which takes care of dependencies on past message symbols, e.g. due to syntax and/or semantics of the H.264/AVC CAVLC. The channel term in $\log \Pr(w_1^k|\underline{y})$ is clearly additive; its summands will have to be conditioned on $w_1^{i-1}$, since the choice of VLC codebook for $w_i$ may depend on past symbols.

The *a priori* probabilities $P(w)$ must be known in order to compute this MAP metric. Assuming that the compression is efficient, the probability of emitting a codeword will be exponentially related to its length:

$$P(w) = \frac{2^{-\ell(w)}}{\sum_w 2^{-\ell(w)}}. \tag{2.14}$$

It can be seen that by this assumption, (2.13) reduces to the maximum likelihood (ML) metric.

A key difference to sequential decoding of convolutional codes is the fact that not all syntactically valid paths correspond to valid decodings of a packet, since the header information imposes additional constraints. Only paths that have the correct length in bits *and* encode the correct number of SMBs (in the slice) are valid decoder outputs. This yields some error correction capability, since semantically invalid paths can be eliminated from the decoder stack. Several kinds of redundancy in the encoded stream of residuals (B/C partitions) can be exploited by the sequential decoder to correct some errors:

- mismatch between the actual source and its model in the encoder (e.g. using a memoryless model for a Markov source),
- mismatch between ideal and actual codeword lengths (e.g. integer codeword lengths, unused leaves/codewords in the VLC tree),
- semantic side information about the encoded content.

The first two kinds of redundancy are of little importance in H.264/AVC: on the one hand, the CAVLC syntax has been closely matched to the correlation structure present in the residual data. As a consequence, having more precise a priori source probabilities, whether transmitted as side information or estimated online, yields only minor performance improvements, if at all. Efficient use of statistical model redundancy is also hampered by typical relatively small packet sizes. On the other hand, although there are some unused codewords in some of the VLC tables, they generally differ in just one position from the longest codeword in a given table; thus they are unlikely to occur as result of a bit error, which would otherwise be detected (cf. Section 2.2). In summary, little redundancy is left in the residuals data stream that could be easily exploited by the decoder to correct the errors; the statistical properties of the encoder output are close to those of a binary symmetric source. These observations together with the analysis in Section 2.2 suggest that the main source of redundancy is the semantic side information contained in the A packets, that is, outside the actual residual data stream in the B/C packets. That information puts semantic constraints on the contents of the residual packets. Of these constraints, the simplest to exploit in a sequential decoder is the knowledge of the number of (S)MBs that are encoded in a given packet of length $n$.

### 2.6.2 Additional Synchronization Points

The side information sent out-of-band (cf. Section 2.5) can also be used to signal additional synchronization points to a sequential decoder. The boundaries of MBs in the code stream are natural candidates for resynchronization, since the decoder already checks for the correct number of MBs at the end of a packet. The decoder is informed of the current bit position every $m$th MB in a slice, i.e. it knows the length $n$ (in bits) and the position of a segment of the code stream encoding $m$ MBs (or less, at the end of a slice). Note that this is not the same as having smaller packets of at most $m$ macroblocks, due to the prediction mechanism that causally affects the encoding of MBs within a slice. The sequential decoder uses the extra side information to simply discard paths that do not line up correctly at the synchronization points.

Providing a synchronization point every $m$ MBs has two shortcomings: first, the rate of side information is still limited by its granularity on the MB level (possibly SMB). This could be problematic for video encoded at high rates with a small quantization parameter, resulting in large intervals between (S)MB boundaries in the code stream. However, that is unlikely to be a problem in low to medium rate wireless applications. The other shortcoming is more serious: increasing the frequency of synchronization points does not necessarily increase the capability of correcting (or detecting) VLC errors, regardless of decoder complexity. For example, it cannot prevent confounding two codewords of the same length. Furthermore, errors in the FLC fields of the code stream still remain completely unnoticed. Fortunately, both problems can be mitigated by proper postprocessing — impairment detection and error concealment.

### 2.6.3 Postprocessing

The CAVLC for the residuals contains also fixed-length coded fields, for example to code sign and mantissa of coefficient values. The decoding metric assigns uniform probabilities to FLC fields, hence FLC errors remain undetected and will cause some distortion like in the detection based on syntax analysis. A more severe decoding error occurs when the sequential decoder outputs a wrong VLC path. This is more likely to occur within I frame slices since these will utilize the maximal packet size. Moreover, decoding a wrong VLC path causes a larger distortion for I frames than for the P and B frames. The distortion in the P and B frames is in prevailing majority of the cases negligible small due to the efficiency of the temporal prediction and using of data partitioning. If no scene change occurs, in natural scene video sequences the residuals of P and B frames are usually rather small. Using data partitioning, motion vectors are correctly received and thus, the distortion caused by wrong residuals is very small. Spatial prediction used for I frames has lower efficiency than the temporal prediction and therefore, the residuals have usually higher values. This leads to considerable visual artifacts if the residuals are wrong. Furthermore, the blocks that use the first block with wrong residuals as a reference for decoding will also be wrongly decoded, that even worsens the situation. Spatial error propagation will result in block-shaped artifacts over several MBs, differing in color and/or intensity from the rest of the picture.

Such decoding errors typically result in impairments corresponding to those caused by decoding of the entire slice out of a damaged packet. Hence, they can also be detected by means of an impairment detection as proposed in Section 2.3.1. Due to data partitioning, for P and B frames there are no errors in motion vectors. As the errors in P and B frame residuals have typically a very small visual importance, it does not make sense to perform an artifact detection here.

For FLC errors, the soft (log-likelihood ratios, LLR) inputs are used to compute an estimate of the expected distortion, which is input to the voting-based impairment detection procedure. After detecting the errors, boundary matching temporal error concealment (cf. Section 3.2.1.3) is used for I frames. Factors that make the temporal error concealment more difficult, such as scene cuts, transitions and fast zooming in/out are not present in the tested “foreman” video sequence.

### 2.6.4 Performance

All simulations are based on the “foreman” sequence in QCIF resolution, encoded with JM using data partitioning. Type A and IDR packets are protected by a stronger channel code than type B/C, so that one can assume that all A and IDR packets are received without error. However, the weaker (or absent) protection of B/C packets causes them to be received with random errors. It is assumed that the physical layer provides the soft information (LLR) for the bits in these packets.

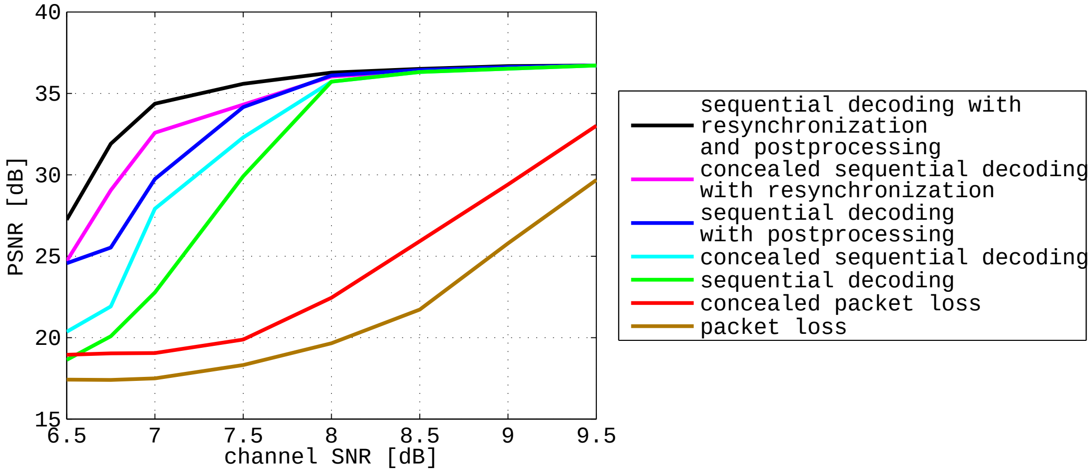


**Figure 2.26**: Comparison of all presented decoder configurations.

The frame rate of "foreman" was reduced by two resulting in 200 frames of video at 15 f/s. The video was encoded with QP = 30, GOP of size 10 frames without B frames, maximal packet size 750 bytes, and then transmitted over a binary-input additive white Gaussian noise (BIAWGN) channel (cf. Appendix D.1.2). The total file size was 169509 bytes (corresponding to about 100 kbit/s), of which 69197 were in error-free A packets. The remaining 100312 bytes were in B/C packets and were decoded with the sequential decoder (in average 10 different realizations of noise were taken into account per $E_b/N_0$ point).

Synchronization points were added with a frequency of $m_I = 4$ in I frames and $m_P = 20$ in P frames, resulting in the same average side information rate of 2% in both frame types, or 15837 bits in total. To model error-free transmission of this information, 3 × 40 bytes of headers were added and assumed a channel code of rate 0.9, which in the practical decoder operating region above 6 dB $E_b/N_0$ is at more than 3 dB from capacity. All together, this corresponds to an increase of 0.1 dB in $E_b/N_0$ compared to a reference system without additional synchronization information. To make comparisons possible, this shift is included in the plots shown. The computational complexity of sequential decoding becomes exponential when operating above the cut-off rate. Therefore, the complexity was limited by letting the decoder drop at most 100 invalid paths before declaring a slice *erasure*. Some more details to the performance of sequential decoder can be found in [69].

**Figure 2.26** depicts the YUV-averaged sequence PSNR in dependency of the channel $E_b/N_0$. Results are shown for different decoders and different postprocessing scenarios. Lost or erased packets (slices) are either left as is or concealed with copy-paste from the previous frame. The best results are obtained when combining the sequential decoder with the artifact detector that generates concealment requests. The additional synchronization information results in about 0.1–0.2 dB gain in $E_b/N_0$ (after the 0.1 dB penalty). The quality improvement resulting from using the resynchronization is for the same rate about 5 dB for $E_b/N_0 = 7$ dB. The postprocessing adds another improvement of 2 dB.

**Figure 2.27** shows the PSNR evolution in time for $E_b/N_0 = 7$ dB. Clearly, in this case both the impairment detection and the resynchronization improve the quality perceived by the user considerably.

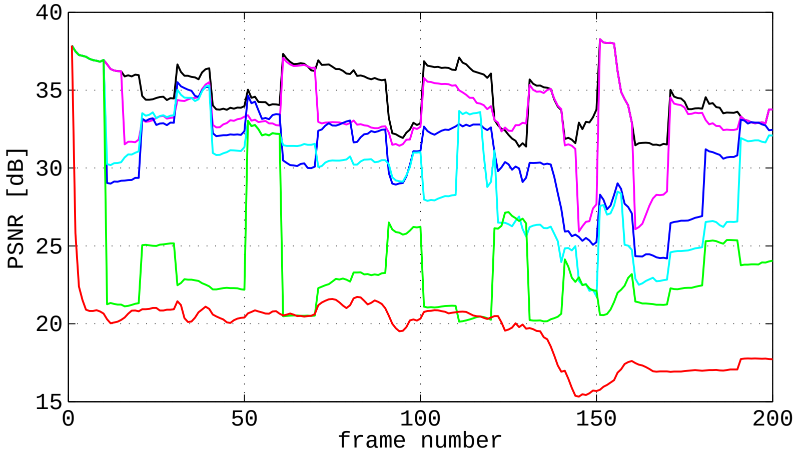


**Figure 2.27**: PSNR evolution in time for $E_b/N_0 = 7\,\text{dB}$. (For legend see **Figure 2.26**.)

The difference to the concealed sequential decoding reaches up to 10 dB.

**Figure 2.28** illustrates the resulting visual gains for $E_b/N_0 = 6.75\,\text{dB}$. The resynchronization decreases the occurrence of the large visual artifacts considerably. The impairment detection further reduces the visually apparent impairments in both cases.

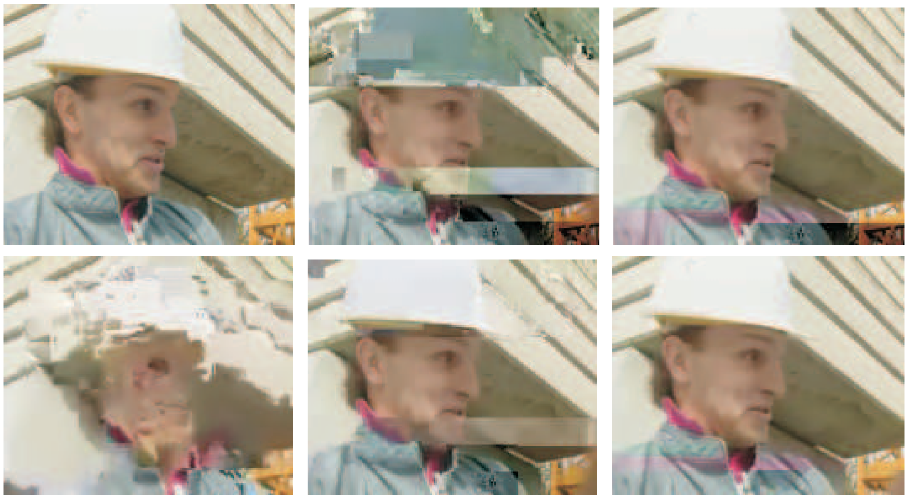


**Figure 2.28**: Screenshots from the "foreman": original and concealed packet loss (left); sequential decoding without and with postprocessing (middle); sequential decoding with resynchronization, without and with postprocessing (right).

For the practical deployment at power limited mobile terminals, sequential decoding still remains too complex. The assumption of the error-free A packets is a rather strong assumption. On the other hand, the packetloss ratio in a static scenario of UMTS is approximately 0.088%, which is quite low and can be even improved by a stronger channel code (less puncturing). The rate can be gained again by sending the B/C packets uncoded. Other practical problem is the knowledge of the soft channel information, that is typically not available in the architecture of the present wireless mobile systems. In a cross-layer designed systems, however, this is easy to solve, since the physical layer as well as the application layer are implemented in the same network element — terminal. Sequential decoding would be beneficial especially for broadcasting and multicasting applications, where the retransmissions are not efficient.

# Chapter 3

# Error Concealment

## Contents

**3.1 Spatial Error Concealment Methods . . . . . . . . . . . . . . . . . . . . . . 58**
  3.1.1 Weighted Averaging . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 59
  3.1.2 Pixel Domain Directional Interpolation . . . . . . . . . . . . . . . . . . . . 60
    3.1.2.1 Edge Detection . . . . . . . . . . . . . . . . . . . . . . . . . . . . 60
    3.1.2.2 Reconstruction . . . . . . . . . . . . . . . . . . . . . . . . . . . . 61
    3.1.2.3 Support for More Edges . . . . . . . . . . . . . . . . . . . . . . . . 63
  3.1.3 Maximally Smooth Recovery Methods . . . . . . . . . . . . . . . . . . . . . 63
    3.1.3.1 First-order Derivative, Uniform Smoothing (FOD-US) . . . . . . . . . 64
    3.1.3.2 First-Order Derivative, Edge Aware (FOD-EA) . . . . . . . . . . . . . 66
    3.1.3.3 Second-Order Derivative, Uniform Smoothing (SOD-US) . . . . . . . 67
  3.1.4 Projection onto Convex Sets (POCS) . . . . . . . . . . . . . . . . . . . . . 69
    3.1.4.1 Convex Constraints . . . . . . . . . . . . . . . . . . . . . . . . . . 69
    3.1.4.2 Implementation and Enhancement . . . . . . . . . . . . . . . . . . . 71
**3.2 Temporal Error Concealment Methods . . . . . . . . . . . . . . . . . . . . 73**
  3.2.1 Motion Compensation . . . . . . . . . . . . . . . . . . . . . . . . . . . . 74
    3.2.1.1 Spatial Motion Vector Interpolation . . . . . . . . . . . . . . . . . . 74
    3.2.1.2 Temporal Motion Vector Interpolation . . . . . . . . . . . . . . . . . 75
    3.2.1.3 Boundary Matching . . . . . . . . . . . . . . . . . . . . . . . . . . . 75
    3.2.1.4 Block Matching . . . . . . . . . . . . . . . . . . . . . . . . . . . . 76
  3.2.2 Model Based Error Concealment . . . . . . . . . . . . . . . . . . . . . . . 77
**3.3 Comparison of Individual Methods . . . . . . . . . . . . . . . . . . . . . . 81**
  3.3.1 Quality of Reconstruction . . . . . . . . . . . . . . . . . . . . . . . . . . 81
  3.3.2 Possibility of Deployment . . . . . . . . . . . . . . . . . . . . . . . . . . 82
  3.3.3 Computational Complexity . . . . . . . . . . . . . . . . . . . . . . . . . . 83
**3.4 Adaptive Error Concealment Mechanism . . . . . . . . . . . . . . . . . . . 85**
  3.4.1 Scene Change Detection . . . . . . . . . . . . . . . . . . . . . . . . . . . 86
    3.4.1.1 Previous Works . . . . . . . . . . . . . . . . . . . . . . . . . . . . 87
    3.4.1.2 Dynamic Threshold Technique . . . . . . . . . . . . . . . . . . . . . 88
  3.4.2 Decision Tree . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 89
  3.4.3 Performance Evaluation . . . . . . . . . . . . . . . . . . . . . . . . . . . 91

ERROR concealment methods are designated to replace the parts of a video stream corresponding to its lost/erroneus packets. The replacement is typically performed by a sort of spatial and/or temporal *interpolation*. Spatial interpolation utilizes the correctly received (and possibly also lost but already concealed) parts of the same picture. Temporal interpolation relies on the previously received frames (preceding or following the damaged frame in the rendering order).

In this chapter, the most important concepts for spatial and temporal error concealment are introduced and evaluated. They were all implemented in order to investigate their performance in different scenarios and thus to provide a fair comparison. Note that such a comparison is not really possible based only on the published information, because often the authors use different quality measures and experimental conditions (resolution, content, frame rate etc.). Several of the tested error concealment methods were newly proposed as a part of this work, several known approaches were enhanced. Since the suitability of the individual error concealment methods is tightly connected with their particular application, in the second part of this chapter, a scene change detection mechanism is proposed. Its output is utilized as one of the criteria to choose the appropriate method. Finally, an adaptive error concealment mechanism is proposed and its performance is compared to the nowadays used approaches.

## 3.1 Spatial Error Concealment Methods

SPATIAL error concealment utilizes only the information from the processed picture for the reconstruction of the lost area. Thus, it is identical to the concealment of still images. The quality of reconstruction depends strongly on the spatial character of the concealed picture. The spatial character of the picture is given by its edges. In general, natural scene images and cartoons consist of smooth areas separated by edges. A special case of image content is pattern, consisting of some regularly repeating parts. This type of image is typical e.g. for backgrounds and artificial images. Examples of different picture area types are depicted in **Figure 3.1**. The approaches to conceal the missing parts with different spatial character are not necessarily identical. Whereas the smooth areas can be interpolated by simple averaging or by optimization of a smoothness cost function, more sophisticated approaches are necessary to deal with areas containing edges. The smoothing should only be performed along the edges, the edges should be prolonged. The resolution of the image together with the size and shape of the lost area determine the interpolation problem to be solved. The low resolutions, this work is dedicated to, are challenging for the most of the known spatial error concealment methods. This is caused by the lower correlation between the neighboring pixels.

In the following, the key concepts of spatial error concealment are presented. Weighted averaging is the simplest and the mostly used method. Pixel domain directional interpolation includes methods aiming at preserving the edges within a picture. A newly proposed directional interpolation supporting the reconstruction of more edges per missing block is presented. Maximally smooth recovery denotes a class of error concealment methods that formulates the error concealment problem as a problem of minimizing the smoothness cost function. The general idea was improved as a part of this work to recover the edges without blurriness. Finally, an approach using projection onto convex sets was enhanced to support more than one edge and investigated with respect to various implementation settings.

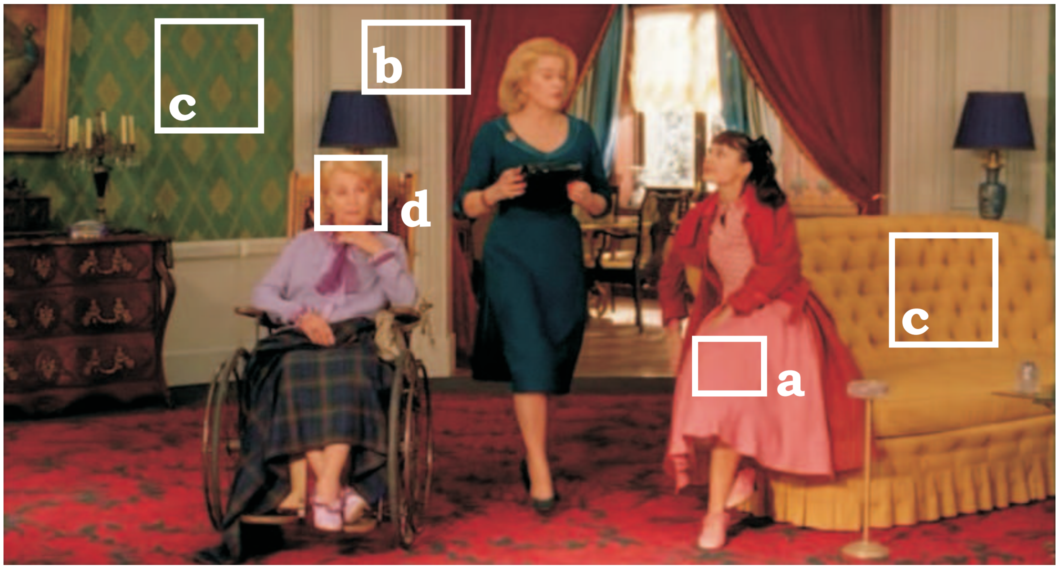


**Figure 3.1**: Examples for smooth area (a), clear edges (b), pattern (c), and an area of a human face, difficult to conceal by spatial error concealment(d).

### 3.1.1 Weighted Averaging

The simplest spatial error concealment method is interpolating of the missing block by an average of the boundary pixel values. Such interpolation, however leads to a monotone block, apparently distinguishable from the rest of the picture, especially if the missing block was not smooth. The *weighted averaging* [48] improves this situation by weighting the contribution of the boundary pixels to their average according to the distance between the interpolated pixel and the opposite boundary. Consider an $M \times N$ large missing block $\underline{\underline{F}}$, consisting of pixel values $f_{i,j}$ (of one color component), with $i$ being the row index and $j$ being the column index. The top left corner corresponds to the $(0,0)$ point. Each pixel value of the missing macroblock is then interpolated as a weighted linear combination of the nearest pixels in the west (w), east (e), north (n), and south (s) boundaries:

$$\widehat{f}_{i,j} = \frac{1}{M+N+2}\left[(N-j+1)b_i^{\mathrm{w}} + jb_i^{\mathrm{e}} + (M-i+1)b_j^{\mathrm{n}} + ib_j^{\mathrm{s}}\right], \tag{3.1}$$

with $b_i^{\mathrm{w}} = f_{i,0}$, $b_i^{\mathrm{e}} = f_{i,N+1}$, $b_j^{\mathrm{n}} = f_{0,j}$, and $b_j^{\mathrm{s}} = f_{M+1,j}$ as shown in **Figure 3.2** (left). Each color component is concealed in the same way. This method only performs well if the missing block is smooth or contains only edges in vertical and/or horizontal directions. Otherwise it produces visible artifacts as can be seen e.g. in **Figure 3.3** (left). Note that if the neighbourhood of the concealed area is not smooth enough, i.e. it is noisy, the weighted averaging results in a grid pattern as can be seen in **Figure 3.3** (right) for the “soccer” sequence.

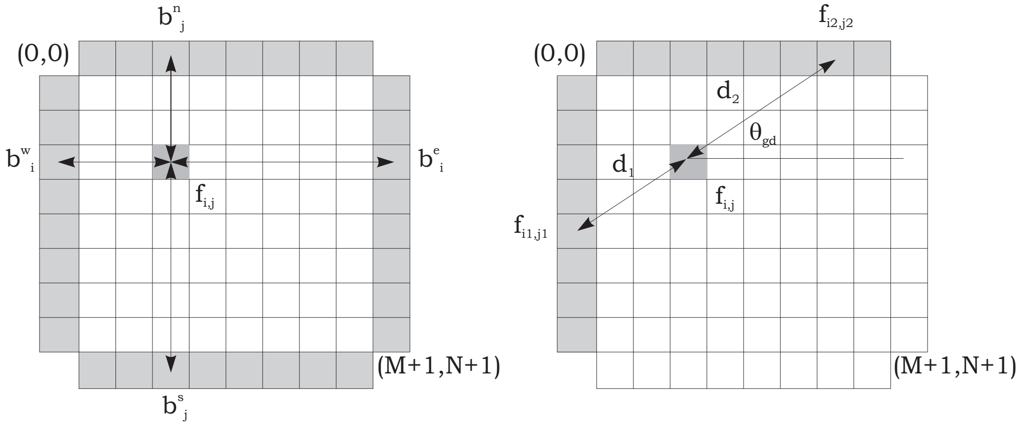


**Figure 3.2**: Spatial interpolation: weighted averaging (left), directional interpolation showed for the whole block (right).

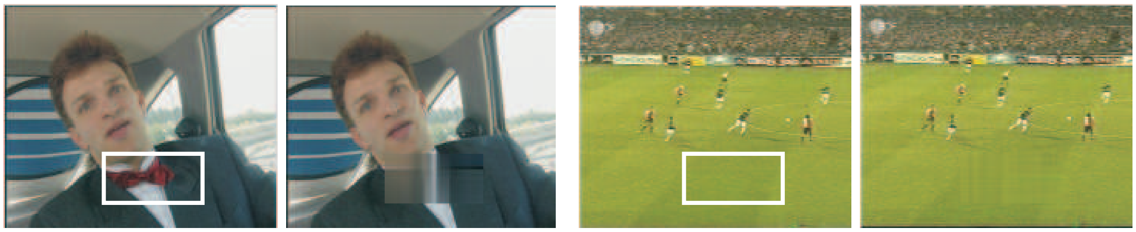


**Figure 3.3**: Performance of weighted averaging: concealment of a missing part in “carphone” and in “soccer” video picture. White rectangle marks the area to be concealed.

### 3.1.2 Pixel Domain Directional Interpolation

The possibilities of spatial prediction type based interpolation are limited by the eight predefined prediction types [7]. A natural enhancement is a pixel domain directional interpolation. In [89] the edge is first classified in one of four main directions and then smoothing in the identified direction is performed. The drawback of this method is, that it supports only one main direction for the whole missing block. While this is possibly sufficient for higher resolution, it brings limited improvement for low resolutions where often more edges meet in one missing area. Therefore, [90] refines the edge direction determination to more directions. In [91], eight directions are distinguished; smoothing is performed recursively in all detected directions, from the outer pixels until filling up the gap (lost part). The edges can be detected in different ways, the one employed in this thesis is described in the following.

#### 3.1.2.1 Edge Detection

The very popular edge detection method is based on the horizontal and vertical Sobel mask given as

$$\underline{\underline{S}}_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}, \quad \underline{\underline{S}}_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}. \tag{3.2}$$

The masks are applied on the luminance value $b_{i,j}$ of each pixel at the missing area boundaries as follows:

$$G_x(i,j) = \mathrm{vec}(\underline{\underline{S}}_x)^T \mathrm{vec}(\underline{\underline{B}}(i,j)), \qquad G_y(i,j) = \mathrm{vec}(\underline{\underline{S}}_y)^T \mathrm{vec}(\underline{\underline{B}}(i,j)), \tag{3.3}$$

with vec denoting the *vectorization* operator (column wise stacking of a matrix into a vector) and $\underline{\underline{B}}(i,j)$ being the relevant part of a boundary:

$$\underline{\underline{B}}(i,j) = \begin{bmatrix} b_{i-1,j-1} & b_{i-1,j} & b_{i-1,j+1} \\ b_{i,j-1} & b_{i,j} & b_{i,j+1} \\ b_{i+1,j-1} & b_{i+1,j} & b_{i+1,j+1} \end{bmatrix}. \tag{3.4}$$

The magnitude of the gradient $|G(i,j)|$ and its direction $\theta_g(i,j)$ can then be calculated for each pixel at the missing block boundary as follows:

$$|G(i,j)| = \sqrt{G_x^2(i,j) + G_y^2(i,j)}, \tag{3.5}$$

$$\theta_g(i,j) = \arctan\left[\frac{G_y(i,j)}{G_x(i,j)}\right]. \tag{3.6}$$

In order to reduce the computational complexity, the edge detection is applied to the luminance component only — the color components are smoother and thus contain less information. The additional utilization of the color components may, however, be beneficial as shown in [92].

The slope $a(i,j)$ of the edge, perpendicular to the gradient direction $\theta_g$, can be expressed as

$$a(i,j) = \frac{G_y(i,j)}{G_x(i,j)} = \cot\left[\theta_g(i,j)\right]. \tag{3.7}$$

The dominant gradient direction within a certain area described by an index set $\mathcal{A}$ can be subsequently determined as a sum of all pixel gradients in $\mathcal{A}$ weighted by their magnitude:

$$\theta_{gd} = \frac{\sum_{\forall i,j \in \mathcal{A}} \theta_g(i,j)|G(i,j)|}{\sum_{\forall i,j \in \mathcal{A}} |G(i,j)|}. \tag{3.8}$$

When detecting the edges it makes sense to set a threshold on the magnitude, that decides what is considered as an edge at all. The threshold has to be determined empirically according to the desired application.

#### 3.1.2.2 Reconstruction

After detecting the main edge direction $\theta_{gd}$ with corresponding edge slope $a_d = \cot[\theta_{gd}]$, the edge is prolonged — the block or its part is interpolated in that direction by means of weighted averaging as illustrated in **Figure 3.2** (right):

$$f_{i,j} = \frac{1}{d_1 + d_2}\left[d_2 f_{i_1,j_1} + d_1 f_{i_2,j_2}\right], \tag{3.9}$$

where $f_{i_1,j_1}$ and $f_{i_2,j_2}$ are the points at the boundaries from which the missing pixel is interpolated. They are obtained as an intersection of the block boundaries with a line of slope $a_d$ crossing the pixel $f_{i,j}$ being concealed. Symbols $d_1$ and $d_2$ denote the distance of $f_{i,j}$ from $f_{i_1,j_1}$ and $f_{i_2,j_2}$ respectively.

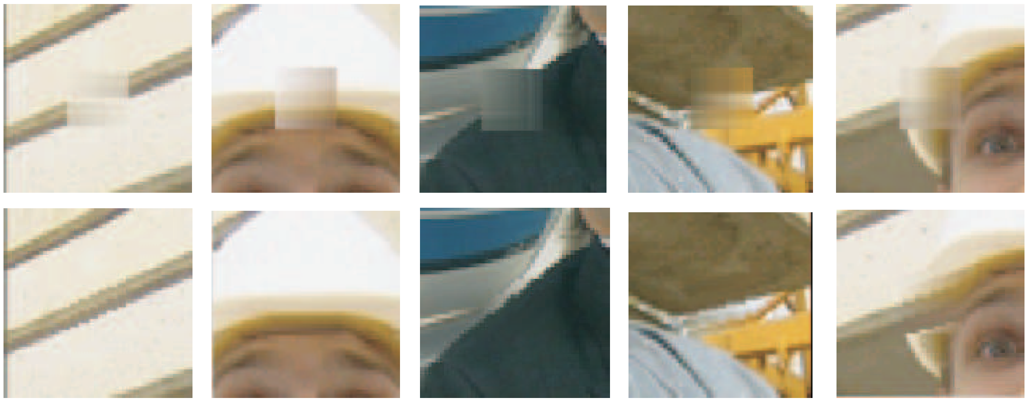


**Figure 3.4**: Reconstruction of a lost macroblock (in the middle of each image) by means of weighted averaging (upper row) and directional interpolation in the main direction (lower row).

| method/image | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| weighted averaging | 31.02 | 32.17 | 30.91 | 31.93 | 35.30 |
| directional interpolation | 42.77 | 46.77 | 38.16 | 34.58 | 32.59 |

**Table 3.1**: PSNR (in dB) of the images in **Figure 3.4** (in the corresponding order from left to right), calculated only for the shown area cut out of the whole video frame.

Note that the indexes $i_1, j_1$ and $i_2, j_2$ are rounded to integer values. Common averaging is a special case of weighted averaging with $d_1 = d_2 = 1/2$. While working well for the images with monotone areas (e.g. cartoons), averaging without weighting can result in blocking artifacts for natural scene videos.

This method performs well for the recovery of parts containing one dominant direction only. However, it does not provide satisfying results if there are more dominant edges entering the missing area, as demonstrated in **Figure 3.4** and in **Table 3.1**. Moreover, if the edges crossing the missing block are not linear, pixel based directional interpolation may cause unpleasant artifacts. All experiments in this section were performed with $3 \times 3$ MB large areas cut from uncompressed pictures corresponding to frames chosen from QCIF video sequences. The macroblock in the middle is assumed to be lost and thus concealed. The PSNR was calculated for the displayed area only (not over the whole picture/video frame).

In the first three reconstructed cases there is a considerable difference between the weighted averaging and the directional interpolation determined in terms of both visually inspected quality and PSNR. However, this is not the case for the reconstruction of the fourth case. If the original picture is not known, it is difficult to find an error in the image reconstructed by directional interpolation, whereas it is clear that the image reconstructed by weighted averaging suffers a distortion. In the fifth case, the PSNR is clearly in favor of weighted averaging. The visual inspection is subjective, but at the first glance, the artifact caused by the directional interpolation might not necessarily be as disturbing as the blurry block interpolated by weighted averaging. To avoid such artifacts, caused by the presence of more dominant edges in the picture, an enhancement of directional interpolation in the pixel domain is necessary.

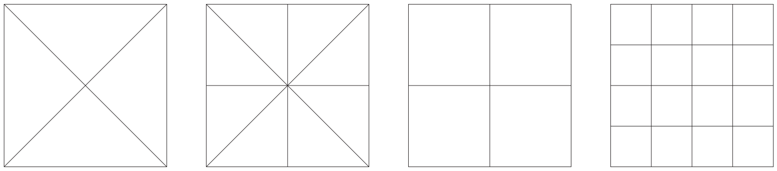


**Figure 3.5**: Fixed size partitioning: four and eight triangular partitions, four and sixteen rectangular partitions.

#### 3.1.2.3 Support for More Edges

In natural scene pictures with resolution as small as QCIF, there are still many blocks with more than one dominant edge. In such case it is important to decide how the edges will be prolonged. Prolonging the edges results in partitioning of the missing block, each partition can be further recovered by smoothing. The method proposed in [93] matches the edges entering and leaving the missing area, partitions the area accordingly and smoothes preserving the edges. Unfortunately, the matching of edges cannot always be performed reliably — especially not if the edges are not linear and if there are too many edges. Furthermore, such method requires the knowledge of all four boundaries.

Therefore in [70], other approaches for partitioning and interpolation were investigated. The missing blocks are first subdivided into *regions of dominance* — each such region supporting one main edge direction. Interpolation is then similar as in the previous section, applied to each particular region separately. Subdividing of the block into equally sized and shaped partitions was shown less efficient than variable size partitioning, performed in following steps:

- Segment missing blocks into M equally sized and shaped partitions.
- Calculate the main direction for each of the partitions.
- Decide according to a predefined threshold, which partitions possess clear edges.
- Attach partitions without clear edges to those with clear edges that correlate most to their detected direction.
- Merge segments with the same dominant direction in a region of dominance.
- Perform pixel domain directional interpolation for each region of dominance.

The shape of the fixed partitions can be chosen either triangular or rectangular as illustrated in **Figure 3.5**. The triangularly shaped partitions have the advantage of having all a “direct access” to the boundary. For the rectangular partitions the merging of regions have to be performed in more steps (e.g. two steps for eight partitions), since only the outer partitions are connected with the boundary. The inner partitions need to be then attached to the outer partitions according to the magnitude of their edges. Thus, in this work the triangular partitioning is considered. Performance of such eight partitions based pixel domain directional interpolation can be seen in **Figure 3.6** and **Table 3.2**.

### 3.1.3 Maximally Smooth Recovery Methods

Another approach, based on solving an optimization problem with a quadratic smoothness cost function was proposed in [94] and enhanced in [95] to the temporal direction. In [96], an improved smoothness cost function is proposed, that considers also a second-order derivative to allow for sharper edge

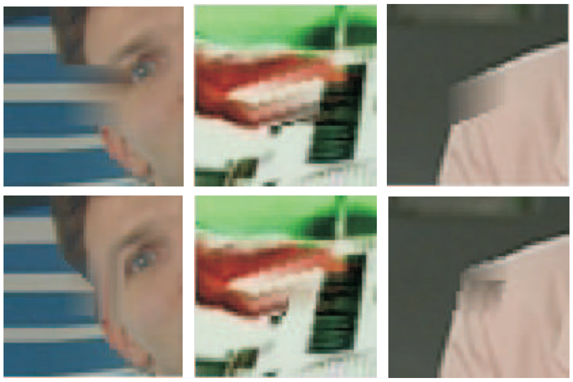


**Figure 3.6**: Reconstruction by directional interpolation with and without segmentation.

| method/image | 1 | 2 | 3 |
|---|---|---|---|
| directional interpolation | 33.61 | 22.30 | 35.11 |
| segmentation | 36.94 | 28.87 | 39.05 |

**Table 3.2**: Comparison of PSNR (in dB) for images in **Figure 3.6**.

reconstruction. The optimization performed in the above mentioned publications was performed generally for the case where some transform domain coefficients were received correctly. Such situation is only possible for layered coding that is not considered in this work. Thus, in the following, the methods are presented for the case where the loss of a macroblock means the loss of all its coefficients.

#### 3.1.3.1 First-order Derivative, Uniform Smoothing (FOD-US)

Let $\underline{\underline{F}}$ represent a block composed of $N$ samples and $f_{i,j}$ the original value of the sample $(i, j)$ in $\underline{\underline{B}}$, where $i$ refers to the row index and $j$ to the column index. The problem is to find the estimates for the lost pixel values $\widehat{f}_{i,j}$ that minimize the smoothness cost function defined as a weighted square difference between adjacent samples in four directions — west (w), east (e), north (n) and south (s).

$$\Psi = \frac{1}{2} \sum_{(i,j)\in\underline{\underline{F}}} \left[ w^{w}_{i,j}(\widehat{f}_{i,j} - \widehat{f}_{i,j-1})^2 + w^{e}_{i,j}(\widehat{f}_{i,j} - \widehat{f}_{i,j+1})^2 + w^{n}_{i,j}(\widehat{f}_{i,j} - \widehat{f}_{i-1,j})^2 + w^{s}_{i,j}(\widehat{f}_{i,j} - \widehat{f}_{i+1,j})^2 \right]. \tag{3.10}$$

The constants $w^{w}_{i,j}, w^{e}_{i,j}, w^{n}_{i,j}$, and $w^{s}_{i,j}$ are called smoothing weights, allowing for reduction/suppression or enhancement of smoothing in the desired direction. Smoothing weights should be imposed between every two adjacent samples across the border of the block in order to propagate the boundary information into the block as shown in **Figure 3.7**. Let external boundary samples $b_{i,j}$ be denoted as

$$b_{m,n} = f_{m,n}, \qquad \text{for } (i,j) \notin \underline{\underline{F}}. \tag{3.11}$$

The notation can now be changed to a matrix notation with

$$\underline{f} = \mathrm{vec}(\underline{\underline{F}}), \quad \underline{b}_d = \mathrm{vec}(\underline{\underline{B}}_d), \tag{3.12}$$

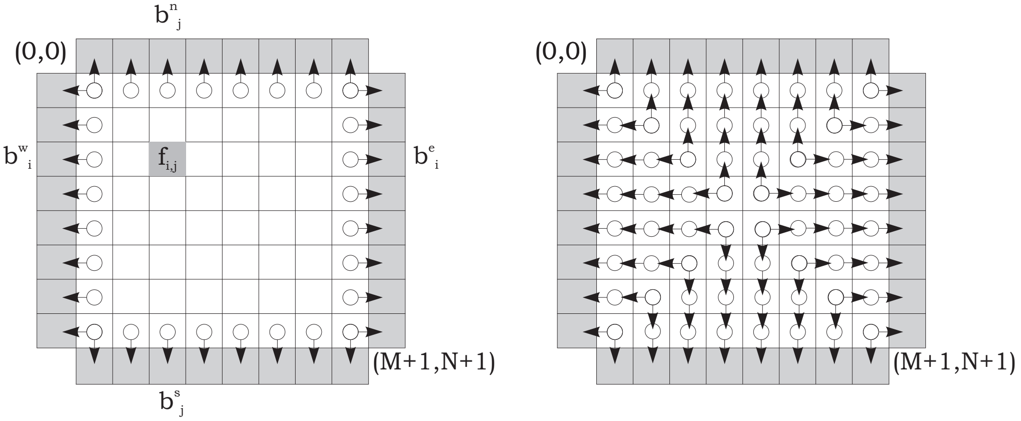


**Figure 3.7**: Spatial interpolation: weighted averaging (left), directional interpolation showed for the whole block (right).

where $\underline{\underline{B}}_d$ is an $M \times N$ matrix composed of zeros and the one-pixel wide boundary in the direction $d \in w, e, n, s$ from the missing block. Thus, $\underline{\underline{B}}_w$ has all elements zero except the first right column containing the west boundary pixel values. Analogically, $\underline{\underline{B}}_n$ has non zero first bottom row, otherwise it is zero. Then also the smoothness cost function (3.10) can be written as

$$\Psi = \frac{1}{2}\left[\|\underline{\underline{S}}_w\widehat{\underline{f}} - \underline{b}_w\|^2 + \|\underline{\underline{S}}_e\widehat{\underline{f}} - \underline{b}_e\|^2 + \|\underline{\underline{S}}_n\widehat{\underline{f}} - \underline{b}_n\|^2 + \|\underline{\underline{S}}_s\widehat{\underline{f}} - \underline{b}_s\|^2\right], \tag{3.13}$$

where matrices $\underline{\underline{S}}_w, \underline{\underline{S}}_e, \underline{\underline{S}}_n$, and $\underline{\underline{S}}_s$ depend on the weights for the smoothing constraints in the four directions. They are rather sparse, for instance, $\underline{\underline{S}}_n$ has all elements zero except the ones in the main diagonal and lower diagonal corresponding to the neighboring pixels the smoothness constraint is imposed to.

To perform the minimization, the derivative of $\Psi$ according to $\underline{f}$ has to be set to zero:

$$\frac{\partial\Psi}{\partial\underline{f}} = \underline{\underline{S}}\,\underline{f} - \underline{b} = 0, \tag{3.14}$$

where

$$\underline{\underline{S}} = \underline{\underline{S}}_w^T\underline{\underline{S}}_w + \underline{\underline{S}}_e^T\underline{\underline{S}}_e + \underline{\underline{S}}_n^T\underline{\underline{S}}_n + \underline{\underline{S}}_s^T\underline{\underline{S}}_s, \tag{3.15}$$

$$\underline{b} = \underline{\underline{S}}_w^T\underline{b}_w + \underline{\underline{S}}_e^T\underline{b}_e + \underline{\underline{S}}_n^T\underline{b}_n + \underline{\underline{S}}_s^T\underline{b}_s. \tag{3.16}$$

By choosing the smoothing constraints properly, the invertibility of the matrix $\underline{\underline{S}}$ can be guaranteed and then the optimally (with respect to the smoothing criterion) reconstructed block $\underline{\underline{F}}_{opt}$ is given by

$$\underline{f}_{opt} = \underline{\underline{S}}^{-1}\underline{b}. \tag{3.17}$$

**Figure 3.7** represents two different approaches to the smoothing — the one shown left imposes the smoothing constraints on the boundary only. The right part of the figure illustrates the smoothing constraints imposed on all neighboring pixels. The performance of the smoothing over all pixels for three different images is demonstrated in **Figure 3.8** and in the **Table 3.3**, compared to weighted

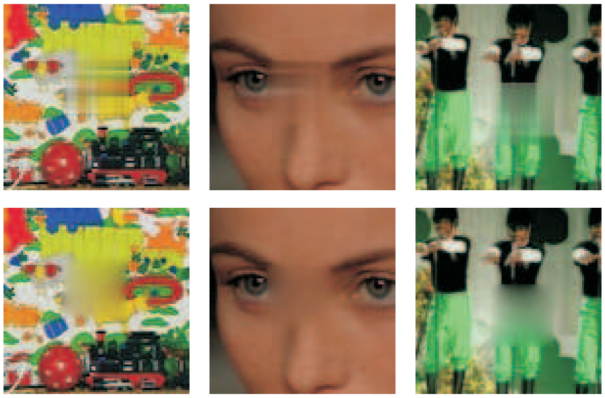


**Figure 3.8**: Images reconstructed by weighted averaging (upper row) and smoothing (lower row).

| method/image | 1 | 2 | 3 |
|---|---|---|---|
| weighted averaging | 38.96 | 41.18 | 37.59 |
| smoothing | 39.17 | 41.19 | 37.39 |

**Table 3.3**: PSNR (in dB) of reconstructed images from **Figure 3.8**.

averaging. There is no essential difference in terms of PSNR performance between weighted averaging and maximally smooth recovery. It is impossible to tell, which of the two methods performs generally better, since it depends on spatial characteristics of the reconstructed image. Weighted averaging performs better for images with vertical and/or horizontal edges, while maximally smooth recovery is slightly better in smooth but not monotone areas. Still, the performance of maximally smooth recovery remains limited since it disregards the edges and smooths also across them. Therefore, in the following, the original maximally smooth recovery is enhanced to support one main direction for smoothing.

#### 3.1.3.2 First-Order Derivative, Edge Aware (FOD-EA)

For directional smoothing, a redesign of matrix $\underline{\underline{S}}$ with respect to the spatial properties of the missing block neighborhood as proposed by the author in [97] is necessary. First the edges have to be detected and the main direction determined. This can be performed e.g. with the method shown in Section 3.1.2.

In the same way as $\underline{\underline{S}}_w \underline{f}$ and $\underline{\underline{S}}_e \underline{f}$ are responsible for horizontal and $\underline{\underline{S}}_s \underline{f}$ and $\underline{\underline{S}}_n \underline{f}$ for vertical differences, the rotation of weights in these matrixes can force directional smoothing in the detected main direction $\theta_{gd} \geq \pi/2$:

$$\underline{\underline{S}}_{en} = |\cos(\theta_{gd})|\, \underline{\underline{S}}_e + |\sin(\theta_{gd})|\, \underline{\underline{S}}_n, \tag{3.18}$$

$$\underline{\underline{S}}_{ws} = |\cos(\theta_{gd})|\, \underline{\underline{S}}_w + |\sin(\theta_{gd})|\, \underline{\underline{S}}_s. \tag{3.19}$$

In the same way, the boundaries have to be transformed:

$$\underline{b}_{en} = |\cos(\theta_{gd})|\, \underline{b}_e + |\sin(\theta_{gd})|\, \underline{b}_n, \tag{3.20}$$

$$\underline{b}_{ws} = |\cos(\theta_{gd})|\, \underline{b}_w + |\sin(\theta_{gd})|\, \underline{b}_s. \tag{3.21}$$

Finally, the solution of such directional smoothing optimization problem is given by

$$\underline{f}_{opt} = \underline{\underline{S}}^{-1}\underline{b}, \tag{3.22}$$

where

$$\underline{\underline{S}} = \underline{\underline{S}}_{en}^T \underline{\underline{S}}_{en} + \underline{\underline{S}}_{ws}^T \underline{\underline{S}}_{ws}, \tag{3.23}$$

$$\underline{b} = \underline{\underline{S}}_{en}^T \underline{b}_{en} + \underline{\underline{S}}_{ws}^T \underline{b}_{ws}. \tag{3.24}$$

Note that for $\theta_{gd} > \pi/2$, instead of east-north and west-south the directions west-north and east-south have to be combined.

The performance of this method can be seen in some examples in **Figure 3.9** and the corresponding **Table 3.4**. In comparison with maximally smooth recovery, the edge aware smoothing causes less blurriness effects. Since in small resolutions, the really smooth areas are rather rare, the edge awareness usually helps the concealed image to look more naturally. Edge aware smoothing achieves good performance particularly for the images with one dominant direction and not as sharp edges.

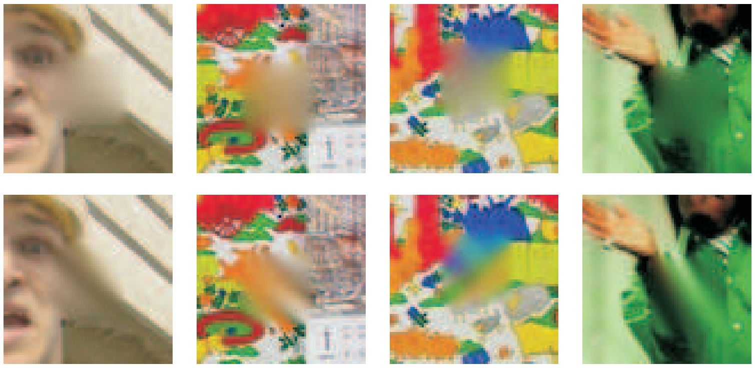


**Figure 3.9**: Images reconstructed by maximally smooth recovery without and with edge awareness.

| method/image | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| FOD-US | 35.60 | 35.23 | 35.50 | 37.18 |
| FOD-EA | 42.04 | 35.51 | 35.94 | 37.62 |

**Table 3.4**: PSNR (in dB) of reconstructed images from **Figure 3.9**.

#### 3.1.3.3 Second-Order Derivative, Uniform Smoothing (SOD-US)

In order to enable a sharper edge reconstruction, [96] proposes an improved smoothness measure $\Phi$ containing also a second-order derivative:

$$\Phi = \sum_{i=1}^{m}\sum_{j=1}^{n}\left(\widehat{f}_{i-1,j} - 2\widehat{f}_{i,j} + \widehat{f}_{i+1,j}\right)^2 + \sum_{i=1}^{m}\sum_{j=1}^{n}\left(\widehat{f}_{i,j-1} - 2\widehat{f}_{i,j} + \widehat{f}_{i,j+1}\right)^2$$

$$+\sum_{i=1}^{m}\sum_{j=1}^{n}\left(\widehat{f}_{i,j}-\widehat{f}_{i+1,j}-\widehat{f}_{i,j+1}+\widehat{f}_{i+1,j+1}\right)^2, \tag{3.25}$$

This can be rewritten into a matrix form

$$\Phi = \|\underline{\underline{A}}_h\underline{\widehat{f}}-\underline{b}_h\|^2+\|\underline{\underline{A}}_v\underline{\widehat{f}}-\underline{b}_v\|^2+2\|\underline{\underline{A}}_d\underline{\widehat{f}}-\underline{b}_d\|^2. \tag{3.26}$$

Matrices $\underline{\underline{A}}_h, \underline{\underline{A}}_v$, and $\underline{\underline{A}}_d$ represent the second order differential operations in horizontal, vertical and diagonal directions, respectively. The vectors $\underline{b}_h, \underline{b}_v$, and $\underline{b}_d$ consist of zeros except at locations corresponding to the boundary samples. The reconstructed block can then be calculated as

$$\underline{f}_{opt} = \underline{\underline{A}}^{-1}\underline{b}, \tag{3.27}$$

with

$$\underline{\underline{A}} = \underline{\underline{A}}_h^T\underline{\underline{A}}_h+\underline{\underline{A}}_v^T\underline{\underline{A}}_v+2\underline{\underline{A}}_d^T\underline{\underline{A}}_d, \tag{3.28}$$

$$\underline{b} = \underline{\underline{A}}_h^T\underline{b}_h+\underline{\underline{A}}_v^T\underline{b}_v+2\underline{\underline{A}}_d^T\underline{b}_d. \tag{3.29}$$

The additional the term $\Lambda$ in $\Phi$ allows for controlling the amount of smoothing applied in the horizontal and vertical directions (as opposed to the two diagonal directions):

$$\Lambda = \sum_{i=1}^{m}\sum_{j=1}^{n}\left(\widehat{f}_{i+1,j}+\widehat{f}_{i-1,j}+\widehat{f}_{i,j+1}+\widehat{f}_{i,j-1}-4\widehat{f}_{i,j}\right)^2. \tag{3.30}$$

Within the combined smoothness measure $\Phi$, the influence of $\Lambda$ is controlled by the constant $\lambda \in \mathbb{R}$ as follows

$$\Phi_L = \|\underline{\underline{A}}_h\underline{\widehat{f}}-\underline{b}_h\|^2+\|\underline{\underline{A}}_v\underline{\widehat{f}}-\underline{b}_v\|^2+2\|\underline{\underline{A}}_d\underline{\widehat{f}}-\underline{b}_d\|^2+\lambda\,\underbrace{\|\underline{\underline{A}}_L\underline{\widehat{f}}-\underline{b}_L\|^2}_{\Lambda}. \tag{3.31}$$

**Figure 3.10** and **Table 3.5** illustrates the reconstruction of two images by SOD-US with different values of the coefficient $\lambda$. The reconstructed images have different character — the first one (from the "foreman" sequence) contains one clear edge, the second one (from the "mobile" sequence) contains a lot of non-linear edges. Surprisingly, for the "foreman" image the SOD-US provides worse results than FOD-US. The best performing error concealment mechanism is clearly the edge aware smoothing. This is not the case for the "mobile" image. The PSNR differences here are rather small. The SOD-US method performs similarly to the FOD-US and FOD-EA, better with increasing $\lambda$. Note that PSNR does not necessarily reflect the quality perceived by user.

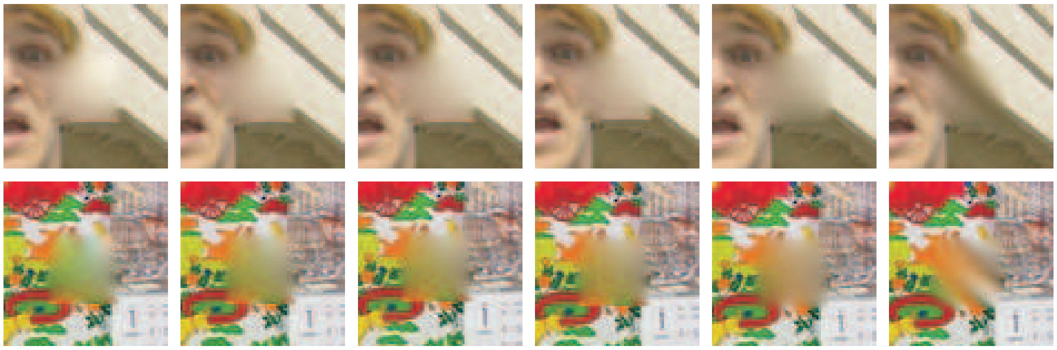


**Figure 3.10**: Images reconstructed by second order derivative smoothing with (left to right) $\lambda = 0, 1, 2, 4$ and by the first order derivative smoothing without and with edge awareness.

| Image | $\lambda = 0$ | $\lambda = 1$ | $\lambda = 2$ | $\lambda = 4$ | FOD-US | FOD-EA |
|---|---|---|---|---|---|---|
| "foreman" | 33.15 | 34.64 | 34.92 | 35.07 | 35.60 | 42.04 |
| "mobile" | 35.17 | 35.46 | 35.53 | 35.60 | 35.23 | 35.51 |

**Table 3.5**: PSNR (in dB) of reconstructed images from **Figure 3.10**.

### 3.1.4 Projection onto Convex Sets (POCS)

In [98] a method was proposed using a larger local neighborhood surrounding the missing block (all eight neighbors). It relays on the a priori properties of typical video images including *consistency* with known values, *smoothness*, and *edge continuity*. According to [99], these desired properties can be formulated as convex constraints.

#### 3.1.4.1 Convex Constraints

**1) The consistency with known values** — the image pixel values belong to the class of signals that takes on a prescribed set of known values: the correctly received sample values should not be changed by the restoration process and the restored values have to be from a known range (typically 0–255). The set $\mathcal{C}_1$ containing all images $\underline{\underline{X}}$ in the $(M \times N)$-dimensional real space $\mathbb{R}^{M\times N}$ with some components equal to known values, can be expressed as

$$\mathcal{C}_1 = \left\{ \underline{\underline{X}} \in \mathbb{R}^{M\times N} : x_{i,j} = k_{i,j},\ (i,j) \in \mathcal{I}_1 \right\}, \tag{3.32}$$

where $x_{i,j}$ is the $(i,j)$th element of image $\underline{\underline{X}}$ and $k_{i,j}$ are known constants in a given index set $\mathcal{I}_1$. The projection operator $\underline{\underline{P}}_1$ onto the convex set $\mathcal{C}_1$ is then given by

$$\left[\underline{\underline{P}}_1 \underline{\underline{X}}\right]_{i,j} = \begin{cases} k_{i,j}, & (i,j) \in \mathcal{B} \\ 0, & (i,j) \in \mathcal{F} \ \text{ and } \ x_{i,j} < 0 \\ 255, & (i,j) \in \mathcal{F} \ \text{ and } \ x_{i,j} > 255 \\ x_{i,j}, & \text{otherwise.} \end{cases} \tag{3.33}$$

Here, the index set $\mathcal{F}$ contains all indexes that correspond to the missing picture area and index set $\mathcal{B}$ includes the indexes of the surrounding blocks. If the value of the component is known, then this value is assigned to the projection; otherwise, it remains unchanged or clipped to the lower/upper limit.

**2) The smoothness and the edge continuity** — the smoothness of natural scene images can be formulated as constraints on spectrum class of signals that takes on a prescribed set of transform coefficients. Such set $\mathcal{C}_2$ contains all images $\underline{\underline{X}}$ in the $(M \times N)$-dimensional complex space $\mathbb{C}^{M\times N}$ with some transform coefficients equal to known values and can be expressed as

$$\mathcal{C}_2 = \left\{ \underline{\underline{X}} \in \mathbb{C}^{M\times N} : \left[\underline{\underline{T}}\,\underline{\underline{X}}\right]_{i,j} = a_{i,j},\ (i,j) \in \mathcal{I}_2 \right\}, \tag{3.34}$$

where $\underline{\underline{T}}$ is a linear transform operator, $\left[\underline{\underline{T}}\,\underline{\underline{X}}\right]_{i,j}$ is the $(i,j)$th transform coefficient, and $a_{i,j}$ are known constraints in a given index set $\mathcal{I}_2$. The projection operator $\underline{\underline{P}}_2$ onto the convex set $\mathcal{C}_2$ is adaptive to the local image characteristics. In monotone/smooth areas of the image, the spectrum has a very low bandwidth. Hence, for missing blocks classified to belong to monotone/smooth areas, the constraint requires any feasible restoration to have a lowpass bandlimited spectrum that results e.g. in projection operator $\underline{\underline{P}}_{2,LP}$ as follows.

$$\left[\underline{\underline{T}}\,\underline{\underline{P}}_{2,LP}\underline{X}\right]_{i,j} = \begin{cases} 0, & \sqrt{i^2+j^2} > r \\ \left[\underline{\underline{T}}\,\underline{X}\right]_{i,j}, & \text{otherwise,} \end{cases} \tag{3.35}$$

where $\underline{\underline{T}}$ is e.g. the discrete Fourier transform operator. The radius $r$ is a threshold specifying the low-pass cutoff frequency. Note that here, $\underline{\underline{T}}\,\underline{X}$ corresponds to the Fourier image with the DC components in the middle (origin of the indexing). Thus, projection $\underline{\underline{P}}_{2,LP}$ sets high-frequency coefficients located outside the bandwidth radius specified by $r$ to zero and leaves low-frequency coefficients unchanged.

In areas of the image containing edges, the spectrum has a bandpass characteristic with the energy localized in transform coefficients that lie in a direction orthogonal to the edge. The rest of the coefficients are small. Thus, for missing blocks classified as edge areas, the constraint requires the restoration to have a bandpass spectrum oriented orthogonally to the detected edge. This leads e.g. to the following operator $\underline{\underline{P}}_{2,BP}$

$$\left[\underline{\underline{T}}\,\underline{\underline{P}}_{2,BP}\underline{X}\right]_{i,j} = \begin{cases} 0, & |i - j \cdot \tan(\theta + \pi/2)| > b \\ \left[\underline{\underline{T}}\,\underline{X}\right]_{i,j}, & \text{otherwise,} \end{cases} \tag{3.36}$$

where $\underline{\underline{T}}$ is again e.g. the discrete Fourier transform operator, $\theta$ ist the angle of the detected edge, and $b$ is a threshold on bandwidth. Thus, $\underline{\underline{P}}_{2,BP}$ sets frequency coefficients located outside the bandpass bandwidth specified by $b$ to zero and leaves the others unchanged.

Both convex projections $\underline{\underline{P}}_1$ and $\underline{\underline{P}}_2$ are involved in the iterative restoration process illustrated in **Figure 3.11**. The damaged MB with its surrounding correct blocks is used to form a large block. The large block is classified to be monotone or to contain one or more edges. It is transformed by Fourier transformation. The transformation coefficients are filtered by the adaptive filter according to the type of the large block determined by the classifier. The filtered coefficients are used to reconstruct the image using inverse transformation. The reconstructed MB is cut and sent back to the input for the next iteration. Thus, the MB to be restored is forced to satisfy the two convex constraints by

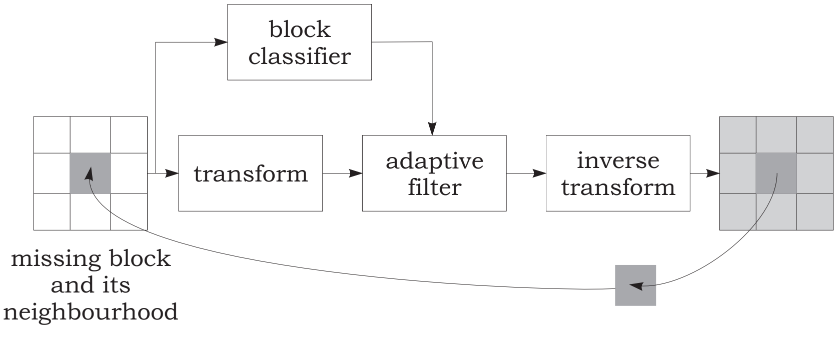


**Figure 3.11**: Block scheme of the iterative POCS image restoration process.

alternatively projecting onto each convex set. The macroblock $\underline{\underline{F}}$ to be restored can be found through the following iteration

$$\underline{\underline{F}}^{(n+1)} = \underline{\underline{P}}_1 \underline{\underline{P}}_2 \underline{\underline{F}}^{(n)}, \tag{3.37}$$

where $n$ denotes the iteration number and $\underline{\underline{F}}^{(0)}$ is the initial guess. The operator $\underline{\underline{P}}_1$ corresponds to copying the restored image macroblock back into the center of the large block. The operator $\underline{\underline{P}}_2$ corresponds to the adaptive filter, which imposes the convex constraint on the values of certain transformation coefficients.

#### 3.1.4.2 Implementation and Enhancement

The implementation of POCS in this thesis was enhanced to support more than one edge. The edge detection was performed by means of the Sobel mask (3.2). The so obtained gradient field was weighted by the distance from the missing block to prefer the edges cutting the block to be concealed. Then the magnitude and the angle of the gradient field was calculated. An voting mechanism was deployed to "quantize" the edges in one of the 32 possible directions. This mechanism simply cumulate the magnitudes of all edges intersecting the missing macroblock. The result is a list of these edge directions ordered by the cumulative magnitude. Only directions of the edges with summed magnitude greater than a predefined threshold are further taken into account when designing the bandpass filter. The initial guess is calculated. Either a lowpass or a bandpass 2D filter is then designed. The bandwidth of the lowpass filter is controlled by a parameter $B_l$, the bandwidth of the bandpass filter is controlled by a parameter $B_b$. The parameter $B_l$ and $B_b$ define the number of frequency samples between the points corresponding to the 3-dB bandwidth of the filter transfer function. The lowpass filter is designed by forcing the frequencies greater than radius $B_l$ to zero. Such rectangular shape of the filter spectrum causes, as a consequence of the Gibbs effect [100], ringing artifacts in the pixel domain. This shape of filter spectrum is what will be further called *rectangular window*. To avoid this effect, the spectrum of the filter can be shaped by another window function. In this implementation, the *Tukey (tapered cosine) window* [103] was chosen, given by

$$W_\alpha(k) = \begin{cases} \frac{1}{2}\left[1+\cos\left(\frac{2\pi(k-1)}{\alpha(N-1)} - \pi\right)\right], & k < \frac{\alpha}{2}(N-1)+1 \\ 1, & \frac{\alpha}{2}(N-1)+1 \le k \le N - \frac{\alpha}{2}(N-1) \\ \frac{1}{2}\left[1+\cos\left(\frac{2\pi}{\alpha} - \frac{2\pi(k-1)}{\alpha(N-1)} - \pi\right)\right], & N - \frac{\alpha}{2}(N-1) < k \end{cases} \tag{3.38}$$

where $\alpha \in [0,1]$ is the ratio of taper to constant sections, $k \in [1, N]$ is the index of the sample and $N$ is the number of samples in frequency between the zero points of the filter transfer function (see **Figure 3.12**). After experimenting with the setting of $\alpha$, the value $\alpha = 0.8$ was chosen, since in the majority of the cases it lead to best quality results. In the same way, the bandpass filter was designed so that it only passes the frequencies corresponding to the edges with the slope perpendicular on the main gradient directions. **Figure 3.13** presents the PSNR of the reconstructed block after a different number of iterations.

**Figure 3.14** shows examples of reconstructed images after 100 iterations using different filter settings and windows. Evidently, the usage of the window plays an essential role in the performance of this error concealment method. The application of the rectangular window leads to ringing artifacts that cannot be compensated for with increasing number of iterations.

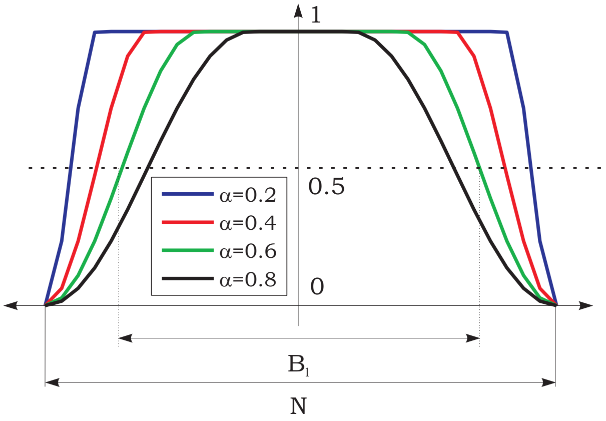


**Figure 3.12**: Tukey (tapered cosine) window function for different values of $\alpha$.

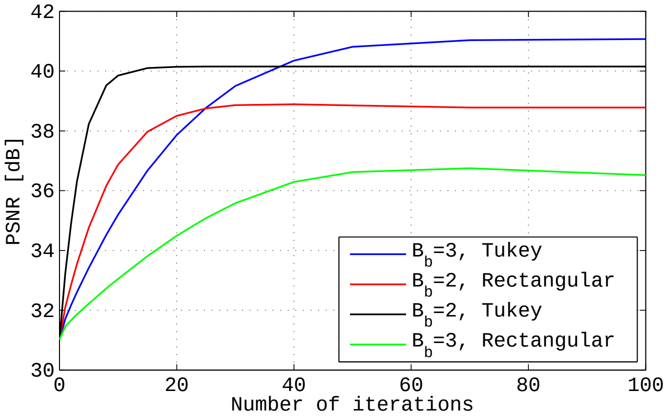


**Figure 3.13**: Quality of reconstruction dependency on the number of iterations for different filter bandwidth and window.

The *initial guess* (0th iteration) has also an influence on the quality of reconstruction and the speed of convergence. In all simulations presented so far, the weighted averaging described in Section 3.1.1 was used. This option was chosen after testing other initial guess possibilities: monotone 50% gray color, boundary pixel averaging, and weighted averaging. The behavior of the reconstruction during 100 iterations for these initial guesses is illustrated in **Table 3.6**. Weighted averaging performs better than the other two methods for lower number of iterations. The averaging without any weighting lead

| Method/Iteration | Initial guess | 1st | 5th | 10th | 25th | 50th | 100th |
|---|---|---|---|---|---|---|---|
| Weighted averaging | 31.02 | 31.71 | 33.42 | 35.20 | 38.77 | 40.81 | 41.07 |
| Averaging | 30.49 | 31.27 | 33.05 | 34.87 | 38.63 | 40.84 | 41.15 |
| 50% gray | 26.11 | 28.07 | 32.36 | 35.47 | 39.37 | 40.94 | 41.23 |

**Table 3.6**: PSNR calculated for the reconstructed regions after 1,5,10,25,50 and 100 iterations using different initial guesses.

**Figure 3.14**: Quality of reconstruction after 100 iterations (from left to right): weighted averaging as initial guess POCS $R = 7$ and $B = 3$ with Tukey window, POCS $R = 7$ and $B = 7$ with rectangular window, POCS $R = 5$ and $B = 2$ with Tukey window, POCS $R = 5$ and $B = 2$ with rectangular window.

to slightly better results after approximately 50 iterations. This is caused by the generic pattern due to the weighting that leads to higher frequency components that cannot be completely suppressed by iterations. It can be concluded that for practical use with very limited number of iterations ($\sim 5$), weighted averaging provides best results. If more iterations are used, filling out by a monotone color can provide better results.

The ability of reconstructing more than one edge is illustrated in **Figure 3.15**. The resulting PSNRs of the reconstruction are 33.61, 33.39, 35.08 and 36.94 dB for unidirectional interpolation, POCS with one edge support, POCS with support of more edges and directional interpolation with segmentation, respectively.

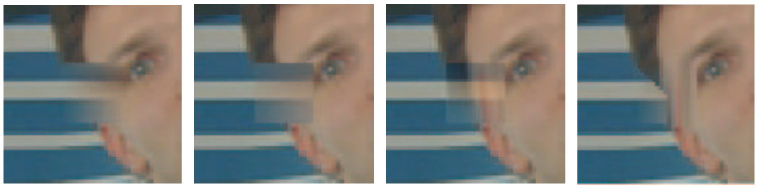


**Figure 3.15**: Reconstruction of a missing macroblock by (left to right): unidirectional interpolation, POCS with six iterations and support for one edge, POCS with six iterations and support for more edges, directional interpolation with segmentation.

## 3.2 Temporal Error Concealment Methods

IN natural low-resolution video sequences, the correlation between two consecutive frames is higher than the correlation of the pixel values within the same frame. Hence, temporal interpolation provides often much better means for error concealment than spatial interpolation.

If the residuals are lost but the motion vectors (MV) have been correctly received, the simplest method is to decode the missing block by setting the missing residuals to zero. This scenario occurs if data partitioning is used. *Decoding without residuals* performs well if the missing residuals are small. On the other hand, the residuals are small if the sequence contains mainly linear motion, which can easily be predicted.

If the whole macroblock information got lost, the simplest method here called the *copy-paste* method (also called zero-motion error concealment) can be used. The missing block is replaced by a spatially corresponding block from the previous frame. This only performs well for low-motion sequences. Better performance is provided by motion compensated interpolation methods. Motion vector estimation methods estimate the motion vector of the missing block from the motion vectors of the neighboring blocks (spatially — within the same frame or temporally — from the previous frames). Boundary and block matching methods do not require the knowledge of any motion vectors, they search the best matching neighbourhood of the missing block in the previous frames. Most of these temporal error concealment methods were already evaluated in [101] for MPEG-2 video sequences with higher resolution. As a part of this work, a refined block matching was proposed. Model based error concealment methods train a model created from statistical properties of a certain object/region of interest. Such model is then used for reconstruction. A method based on principal component analysis was implemented and enhanced in this thesis.

## 3.2.1 Motion Compensation

### 3.2.1.1 Spatial Motion Vector Interpolation

If the motion vectors $\underline{mv}_D$ with $D \in \mathcal{D}$ and $\mathcal{D} = \{\mathrm{n, s, w, e}\}$ of the top (north), bottom (south), left (west), and right (east) neighboring macroblock are known, then the motion vector $\underline{mv}$ of a missing block may be easily approximated by an average of those:

$$\widehat{\underline{mv}} = \frac{1}{|\mathcal{D}|} \sum_{D \in \mathcal{D}} \underline{mv}_D. \tag{3.39}$$

(Here, $|\mathcal{D}|$ is the cardinality of the set $\mathcal{D}$.) The missing block is replaced by the block in the previous frame having the position indicated by $\widehat{\underline{mv}}$. The performance of this method is limited for lower resolutions as each macroblock can also contain parts of different objects moving in different directions. However, H.264/AVC supports motion prediction for blocks of variable size, which can be used to refine the motion estimation [49]. Within this thesis the following refined motion vector interpolation is implemented. The missing macroblock ($16 \times 16$) is first segmented in smaller blocks ($8 \times 8$, $4 \times 4$ or $2 \times 2$). For each such block its MV is estimated as a weighted average of the motion vectors belonging to the nearest neighbouring blocks as depicted in **Figure 3.16**. If no subblocks were used, the motion vectors on the boundary $D$ will be the same for all the blocks. Let $\underline{mv}_D^{(k)}$ be the motion vector of the $k$-th block in the boundary $D$. The estimated motion vector $\widehat{\underline{mv}}^{(i,j)}$ of the block being on the position $(i, j)$ within the missing macroblock can be calculated as a weighted average of block motion vectors at the boundaries:

$$\widehat{\underline{mv}}^{(i,j)} = \frac{d_\mathrm{e}\underline{mv}_\mathrm{w}^{(j)} + d_\mathrm{w}\underline{mv}_\mathrm{e}^{(j)} + d_\mathrm{n}\underline{mv}_\mathrm{s}^{(i)} + d_\mathrm{s}\underline{mv}_\mathrm{n}^{(i)}}{d_\mathrm{e} + d_\mathrm{w} + d_\mathrm{n} + d_\mathrm{s}}. \tag{3.40}$$

This method is directly applicable only for P and B frames. For I frames the motion vectors have to be estimated as described in the following sections. Each missing block is then concealed by the part of the previous picture corresponding to $\widehat{\underline{mv}}^{(i,j)}$.

Motion vector interpolation can lead to blocking artifacts, especially for the video sequences containing inhomogeneous movement and/or non-linear movement. Therefore, in [102] an improved

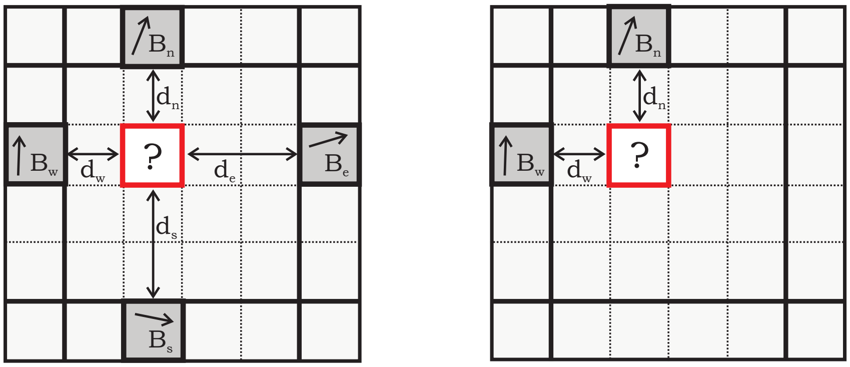


**Figure 3.16**: Refined MV estimation: macroblock segmented to 4 × 4 pixel large blocks. MV interpolation using 4 (left) and 2 (right) neighbouring macroblocks.

method was proposed that sets the weights in (3.40) according to the *side match* criterion. The side match criterion measures the differences between the pixel values at the boundary and within the block obtained by MV interpolation. The parts with higher match get higher weights. In the JM implementation, the side match criterion is used to choose the motion vector — a vector of the block with best side match is chosen.

#### 3.2.1.2 Temporal Motion Vector Interpolation

The motion vectors are not only correlated in the spatial, but also in the temporal domain. Therefore, additionally to the spatial motion vector interpolation or in the cases where the spatially neighboring MVs are not available (e.g. I frames), temporal motion vector interpolation is an option. The efficiency of the temporal MV interpolation depends strongly on the temporal characteristics of the video stream. The performance of this interpolation method decreases considerably with a reduction of frame rate. Utilization of a mean of previous and the following frames can improve the results, but leads to increased delay and storage requirement.

#### 3.2.1.3 Boundary Matching

Let $\underline{\underline{B}}$ be the area corresponding to a one pixel wide boundary (containing the top and/or bottom and/or left and/or right boundary pixels) of a missing block in the $n$-th frame $\underline{\underline{F}}_n$. Motion vectors of the missing block as well as those of its neighbours are unknown (occurs mostly for I frames, but also in specific cases for P frames with some inserted I macroblocks) or not taken into account. The task is to find the coordinates $(\hat{x}, \hat{y})$ of the best match to $\underline{\underline{B}}$ within the search area $\underline{\underline{A}}$ in the previous frame $\underline{\underline{F}}_{n-1}$:

$$(\hat{x}, \hat{y}) = \arg \min_{x,y \in \underline{\underline{A}}} \sum_{i,j \in \underline{\underline{B}}} |\underline{\underline{F}}_{n-1}(x+i, y+j) - \underline{\underline{B}}(i,j)|. \tag{3.41}$$

The sum of absolute differences (SAD) was chosen as a similarity metric for its low computational complexity. The size of $\underline{\underline{B}}$ depends on the number of correctly received neighbors $M$, boundaries of which are used for matching. The area $\underline{\underline{A}}$ is an important design parameter and should be chosen with respect to the amount of motion in the sequence. In the simulations presented in this thesis, $\underline{\underline{A}}$ was a squared area of 9×9 pixels with its center having the same position within $\underline{\underline{F}}_{n-1}$ as $\underline{\underline{B}}$ within $\underline{\underline{F}}_n$. This size was used for concealing the video sequences with its original frame rate. For sequences with reduced frame rates, this size was multiplied by the frame rate decimation coefficient. The macroblock sized area starting at the position $(\hat{x}, \hat{y})$ in $\underline{\underline{F}}_{n-1}$ is taken to conceal the damaged macroblock in $\underline{\underline{F}}_n$. Boundary matching works well if all four boundaries are available and if the motion is linear and homogeneous (all parts moving in the same direction).

#### 3.2.1.4 Block Matching

Better results can be obtained by searching for the best match for the correctly received macroblocks $\underline{\underline{MB}}_D, D \in \mathcal{D}, \mathcal{D} = \{\text{n, s, w, e}\}$, neighbouring the missing one on top (north), bottom (south), left (west), and right (east) side, respectively:

$$[\hat{x}, \hat{y}]_D = \arg \min_{x,y \in \underline{\underline{A}}_D} \sum_{i,j \in \underline{\underline{MB}}_D} |\underline{\underline{F}}_{n-1}(x+i, y+j) - \underline{\underline{MB}}_D(i,j)|, \tag{3.42}$$

$\underline{\underline{A}}_D$ representing the search area for the best match of $\underline{\underline{MB}}_D$, with its center spatially corresponding to the start of the missing macroblock. The final position of the best match is given by an average over the positions of the best matches found for the neighboring blocks:

$$\hat{x} = \frac{1}{|\mathcal{D}|} \sum_{D \in \mathcal{D}} \hat{x}_D; \quad \hat{y} = \frac{1}{|\mathcal{D}|} \sum_{D \in \mathcal{D}} \hat{y}_D. \tag{3.43}$$

The area of MB size starting at the position $(\hat{x}, \hat{y})$ in $\underline{\underline{F}}_{n-1}$ is taken to conceal the damaged macroblock in $\underline{\underline{F}}_n$. To reduce the necessary number of operations, only parts of the neighboring macroblocks can be used for the MV search. For simulations the search block size of $10 \times 8$ was chosen and $\underline{\underline{A}}_D$ two times as large as the searched block.

Similarly to the refined MV interpolation, better results are achieved by searching the motion vectors for such blocks that are smaller than the whole macroblock as shown in **Figure 3.17**. For the search of MVs belonging to a subblock, blocks of the size smaller or greater than the subblock itself may be used. The MVs of blocks belonging to the missing MB can be interpolated using the estimated MVs obtained for the subblocks of the neighbors, using (3.40).

Examples of block matching and boundary matching in comparison with copy-paste error concealment can be seen in **Figure 3.18**. The shown Region Of Interest (ROI) was degraded by one missing macroblock. The resulting PSNR calculated over ROI of the uncompressed "foreman" sequence is 22.75, 24.33 and 25.64 dB for the copy-paste, boundary matching and block matching error concealment, respectively. The PSNR of the "akiyo" sequence reconstructed by copy-paste, boundary matching and block matching is 29.48, 33.28 and 35.71 dB, respectively. Evidently, boundary matching improves the quality of reconstruction considerably compared to the copy-paste mechanism. Block matching with blocks of $8 \times 8$ even more improves the quality of reconstruction. This is caused by better flexibility of the small blocks that allows for compensation of the non-linear and inhomogeneous movement like e.g. the movement of the lips (see "foreman" example).

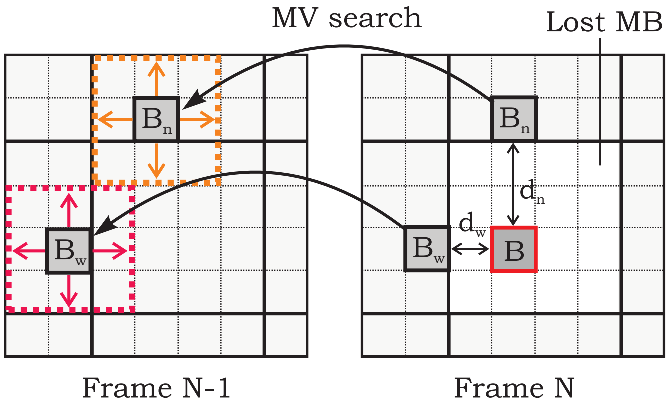


**Figure 3.17**: Refined MV search and estimation: MV search for the nearest 4 × 4 subblocks $\underline{\underline{B}}_{\mathrm{n}}$ and $\underline{\underline{B}}_{\mathrm{w}}$ in the upper and left neighbouring macroblocks (left). Found MVs are used to interpolate the MV of subblock $\underline{\underline{B}}$ belonging to the missing macroblock (right).

### 3.2.2 Model Based Error Concealment

For the sake of completeness, the class of model based error concealment mechanisms has to be investigated as well. The a priori information for error concealment is obtained by training a context-based model for a region of interest and uses the trained model for the reconstruction of lost data within the ROI. The Principal Component Analysis (PCA) has long been used to model the visual content of images, for object recognition (especially faces) and reconstruction [104].

Consider $M$ two-dimensional $p \times q$ training ROI images $\underline{\underline{X}}_i$, $i \in [1, M]$. Each image is transformed into a vector $\underline{x}_i = \mathrm{vec}(\underline{\underline{X}}_i)$ of size $p \cdot q$. The mean image $\underline{m}$ is then estimated by the arithmetic mean over the $M$ ROI images:

$$\underline{m} = \frac{1}{M} \sum_{i=1}^{M} \underline{x}_i \tag{3.44}$$

From all $M$ training ROI images the mean image is then subtracted

$$\underline{\chi}_i = \underline{x}_i - \underline{m}. \tag{3.45}$$

The $(p \cdot q) \times (p \cdot q)$ covariance matrix $\underline{\underline{C}}$ is subsequently estimated by

$$\underline{\underline{C}} = \frac{1}{M} \sum_{i=1}^{M} \underline{\chi}_i \underline{\chi}_i^T = \underline{\underline{A}}\,\underline{\underline{A}}^T, \tag{3.46}$$

where $\underline{\underline{A}} = [\underline{\chi}_1, \ldots, \underline{\chi}_M]$. Determining the eigenvectors $\underline{u}_k$ out of a $(p \cdot q) \times (p \cdot q)$ large covariance matrix $\underline{\underline{A}}\,\underline{\underline{A}}^T$ is not really feasible. If the number of data points $M$ (in this case umber of ROI images) in the image space is smaller than the dimension $(p \cdot q)$ of the space, there will be only $M - 1$ meaningful eigenvectors instead of $(p \cdot q)$. The remaining eigenvalues will have the associated eigenvalues equal to zero. Thus, the solution can be performed by finding the eigenvectors of the $M \times M$ matrix $\underline{\underline{A}}^T \underline{\underline{A}}$ instead: The eigenvalues $\lambda_k$ and the eigenvectors $\underline{v}_k$ of $\underline{\underline{A}}^T \underline{\underline{A}}$ for $k = 1, \ldots, M - 1$ can be found by solving the equation

$$\underline{\underline{A}}^T \underline{\underline{A}}\,\underline{v}_k = \lambda_k \underline{v}_k. \tag{3.47}$$

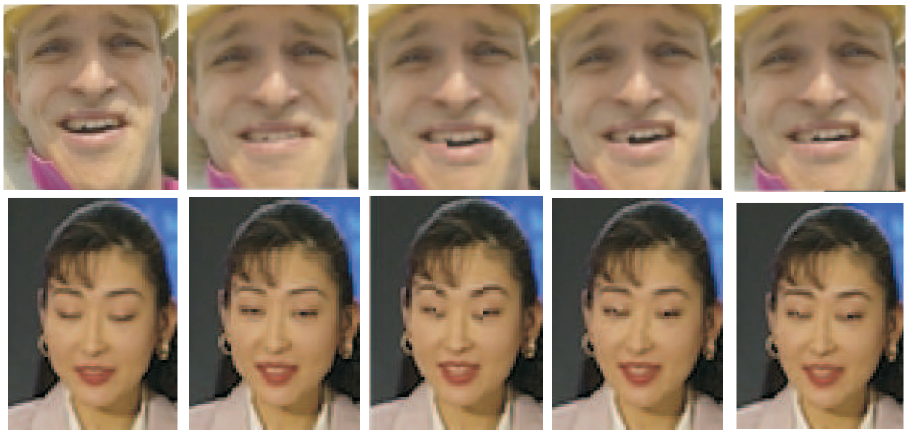


**Figure 3.18**: Screenshots of the video sequences “foreman” (upper row) and “akiyo” (lower row): correctly received previous ROI $\underline{\underline{R}}_{n-1}$, original ROI $\underline{\underline{R}}_n$, reconstructions by copy-paste, boundary matching, and block matching with 8 × 8 blocks.

Multiplication from left by $\underline{\underline{A}}$ yields

$$\underline{\underline{A}}\,\underline{\underline{A}}^T\underline{\underline{A}}\,\underline{v}_k = \lambda_k\underline{\underline{A}}\,\underline{v}_k, \tag{3.48}$$

resulting in eigenvectors $\underline{u}_k$ of $\underline{\underline{A}}\,\underline{\underline{A}}^T$ given by

$$\underline{u}_k = \underline{\underline{A}}\,\underline{v}_k. \tag{3.49}$$

The eigenvectors $\underline{u}_k$ represented as a ROI image $\underline{\underline{U}}_k = \mathrm{vec}^{-1}(\underline{u}_k)$ are called *eigenfaces*. **Figure 3.19** shows the mean image and the eigenfaces obtained from 10 luminance ROI images cut out of the frames of the “foreman” video sequence.

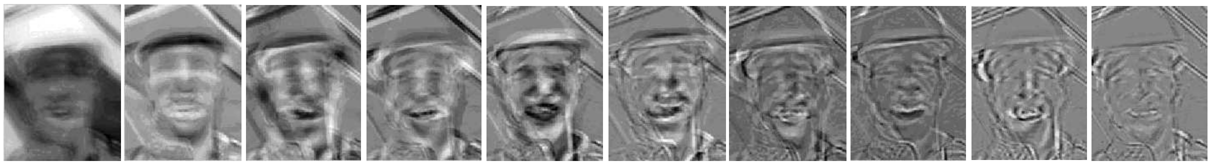


**Figure 3.19**: Mean image $\underline{\underline{M}}$ and eigenfaces $\underline{\underline{U}}_k$ obtained from 10 successive luminance ROI images of the sequence ‘foreman’.

An arbitrary image $\underline{\underline{F}}$ can then be transformed into the basis formed by these eigenfaces. The resulting transformation domain coefficients are given by

$$\gamma_k = \underline{u}_k^T(\underline{f} - \underline{m}) \tag{3.50}$$

for $k = 1, \ldots, M$ and $\underline{f} = \mathrm{vec}(\underline{\underline{F}})$. The so obtained vector $\underline{\gamma} = [\gamma_1, \ldots, \gamma_M]$ describes the contribution of each eigenface. Analogically, the original image can be reconstructed by the transformation domain coefficients:

$$\widehat{\underline{f}} = \underline{\underline{U}}\,\underline{\gamma} + \underline{e}, \tag{3.51}$$

where $\underline{\underline{U}} = [\underline{u}_1, \ldots, \underline{u}_M]$ and $\underline{e}$ denotes the reconstruction error vector. If $\underline{f}$ belongs to the *subspace* formed by the eigenfaces, then $\underline{e} = \underline{0}$ and $\widehat{\underline{f}} = \underline{f}$.

In [105], such PCA model is used in an iterative way for error concealment of the spatially predicted frames of video sequences[1]. The principle of the method is illustrated in **Figure 3.20**. The PCA model (given by the set of eigenfaces) is formed during the runtime of the video sequence based on the already received correct ROI images. Better than estimating the mean image $\underline{m}^{(n)}$ and the covariance matrix $\underline{\underline{C}}^{(n)}$ in each step $n$ (at each new frame) by (3.44) and (3.46), they are estimated recursively as follows

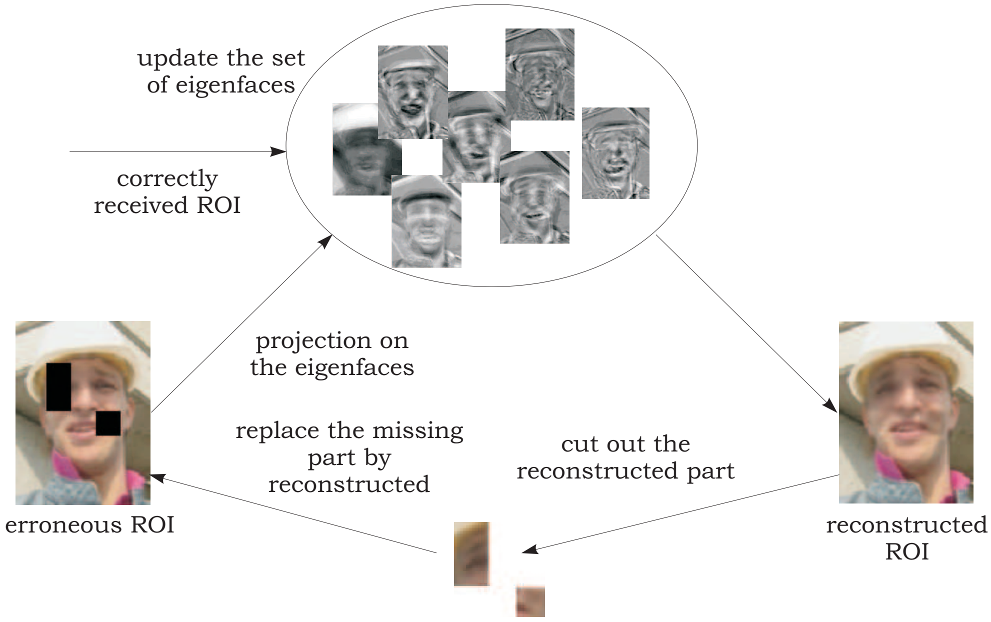

update the set of eigenfaces
correctly received ROI
projection on the eigenfaces
replace the missing part by reconstructed
cut out the reconstructed part
erroneous ROI
reconstructed ROI

**Figure 3.20**: Functional scheme of the iterative reconstruction using a PCA model.

$$\underline{m}^{(n)} = \alpha \underline{m}^{(n-1)} + (1-\alpha)\underline{x}_n, \tag{3.52}$$

$$\underline{\underline{C}}^{(n)} = \alpha \underline{\underline{C}}^{(n-1)} + (1-\alpha)\left[(\underline{x}_n - \underline{m}^{(n)})(\underline{x}_n - \underline{m}^{(n)})^T\right], \tag{3.53}$$

where the parameter $\alpha \in [0,1]$ controls the contribution of the ROI $\underline{x}_n$ received in the step $n$ with respect to the previously received ROIs. A ROI received with some parts missing/distorted is first transformed to the model domain by (3.50), resulting in the coefficient vector $\underline{\gamma}$ and then transformed back by (3.51). By transforming back, the missing part of picture is also reconstructed using the model eigenfaces. This reconstruction can be further refined by applying POCS (cf. Section 3.1.4) — the reconstructed missing part of ROI is cut out, inserted into the original correctly received part of ROI and the whole process is repeated iteratively. The method converges much faster if the missing part of ROI is first concealed (initial guess) e.g. by boundary matching.

Here, it is assumed that the position of ROI is known in the first frame. In practice, the ROI would need to be recognized first, e.g. using a clustering-based picture segmentation [106]. Note that

---

[1] The authors of [105] assign their method to the group of spatial error concealment methods. Since the method needs previous frames for training of the PCA model (even if not directly for the error concealment), in this thesis it was considered a temporal error concealment method.

the ROI can change its position from frame to frame. In order to train the PCA model appropriately, the ROI has to be tracked. In this thesis a simple but efficient ROI tracking method was employed: a several pixel wide inner ROI boundary is searched for in the new frame. The ROI in the position of the best matching boundary (in terms of SAD) is then chosen.

The example in **Figure 3.21** demonstrates that a PCA based model does not necessarily perform better than boundary matching. If the error pattern consists of a large continuous area (upper row), then the PCA based error concealment performs better (ROI PSNR=25.27 dB) than boundary matching (ROI PSNR=23.23 dB). This scenario corresponds to typical slicing that limits the number of MBs or bytes per slice. If the error pattern consists of individual erroneous blocks having all neighbours intact (lower row), then boundary matching (ROI PSNR=29.05 dB) outperforms model based error concealment (ROI PSNR=26.44 dB). These results were obtained using boundary matching as an initial guess. Eight iterations and five previous frames were used for PCA calculation. Boundary matching performs well if boundaries are short and known. The PCA model can worsen the boundary matching initial guess due to shifts of the reconstructed ROI, depending on the previous frames the PCA model was based on. The number of iterations necessary for POCS is rather small. For the upper error pattern in **Figure 3.21** the ROI PSNR after one iteration was 25.21 dB, after two iterations it was 25.26 dB, after 15 iterations 25.27 dB and this value did not change (up to 300 iterations were tested). If the initial guess is weighted averaging (ROI PSNR=14.68 dB), then the error concealment using PCA results in ROI PSNR of 20.24 dB after one iteration, 20.47 dB after two iterations, 20.66 dB for eight iterations and 20.70 dB after 12 iterations which, again does not further change (within the four decimal positions). **Figure 3.22** shows the performance of the PCA model based error concealment in dependence on the number of eigenfaces used for the PCA model. The figure was obtained for the ROI of the “foreman” video sequence and error patterns shown in **Figure 3.21**. The optimum number of frames to train the model depends on the amount of movement in the sequence. In the shown example, four frames already show sufficiently good results. The number of frames determines also the size of a matrix for which the eigenvectors are searched for and thus, the computational

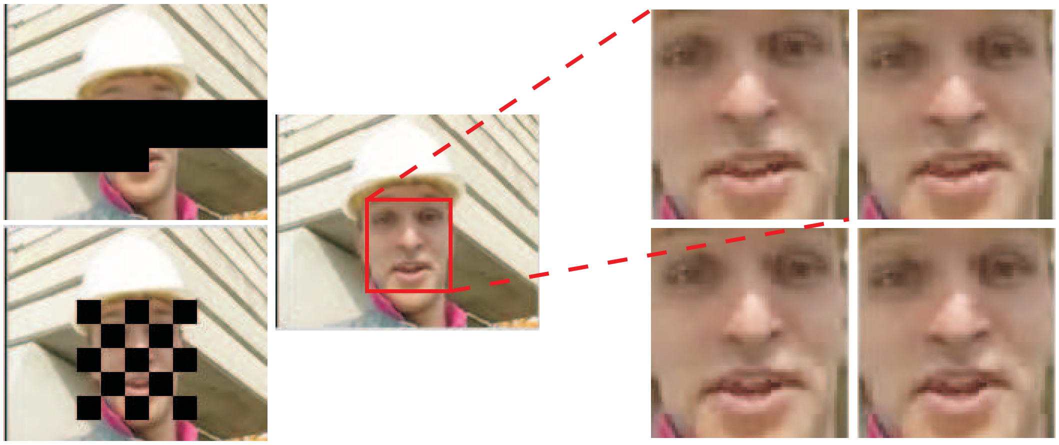


**Figure 3.21**: Frame with error pattern, ROI, reconstructed of ROIs. The ROIs for the given error pattern were reconstructed by boundary matching (left) and a PCA model with POCS (right) using eight iterations and five eigenfaces.

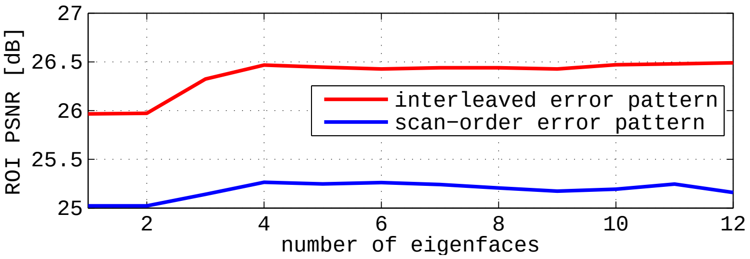


**Figure 3.22**: The quality of reconstruction in dependency on the number of PCA model eigenfaces.

complexity.

## 3.3 Comparison of Individual Methods

In order to choose the best method for a given scenario, the individual error concealment methods have to be compared. In this section a comparison of the above described methods is presented. The comparison was performed in terms of possibility of deployment, quality of reconstruction, and estimation of complexity.

The literature about error concealment usually lacks fair comparison of these methods. Typically, a newly proposed method is only compared to a chosen state-of-the-art technique. This is partly caused by the amount of existing approaches and the difficulty of implementing them all for the sake of comparison.

The only fair comparison is using the reference video sequences and compare the results for the chosen reconstruction quality metric, commonly the PSNR. However, the results have to be taken carefully, since they can differ depending on how the PSNR is calculated (for which colorspace, the way of averaging) and on the encoder settings. Moreover, the complexity is sometimes difficult to evaluate since it may depend on the actual implementation.

### 3.3.1 Quality of Reconstruction

In **Figure 3.23** the seven cases chosen for the comparison of the individual error concealment methods are presented. The cases were chosen from the test video sequences with original or reduced frame rate (by four). Note that the cases 3 and 4 correspond to a scene cut.

**Table 3.7** summarizes the performance of the presented individual error concealment methods under the assumption that all information necessary for the methods is available (MVs, all four neighbors etc.).

Note that the performance of all methods based on the assumption of correctly received neighbors decreases rapidly if not all neighbors are known/correctly received. Moreover, one MB is a sufficiently small area such that spatial error concealment methods still perform well. The results show that after a scene change, spatial error concealment performs clearly better than temporal error concealment. Among spatial error concealment methods, the (segmented) directional interpolation performs well for cases with clearly distinguishable edges (cases 1, 3 and 6). These are also the scenarios where the

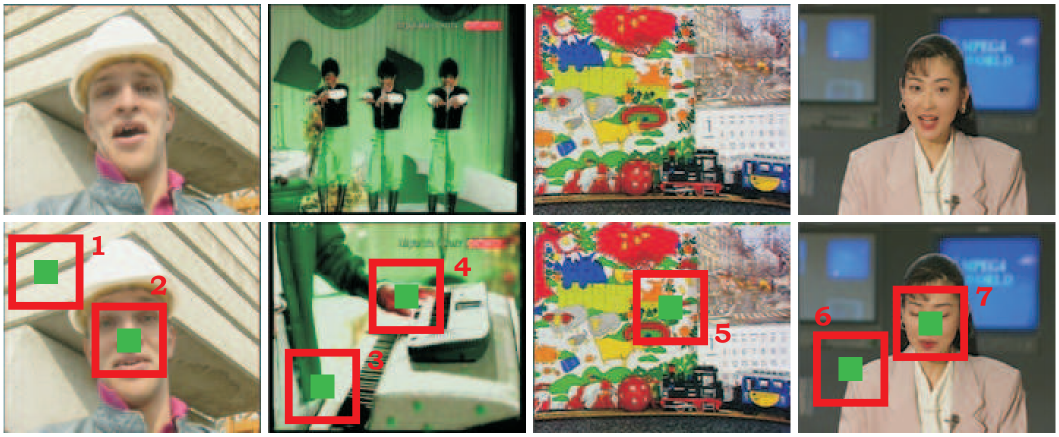


**Figure 3.23**: Cases for performance analysis of individual error concealment methods. Lower row: missing macroblock (green square) and its neighborhood (red square). Upper row: previous frame used for temporal error concealment.

POCS performs well. In other cases (more and/or non-linear edges), the smoothing based methods and weighted averaging cause fewer artifacts, and are thus more suitable. The novel FOD-EA in most cases overperforms both the FOD-US and SOD-US.

From the temporal error concealment methods, the boundary matching and the block matching result in smallest distortion. Motion vector interpolation with side match performs similarly if no scene cut occurs. In cases 3 and 4, the boundary and block matching performed even better due to high motion vector values, the average of which resulted in the choice of area with low similarity to the original.

### 3.3.2 Possibility of Deployment

In practice, the surrounding of the missing block is not always completely known. This is the case for instance for missing blocks at the image boundaries. Another example is the concealment of a missing slice performed per macroblock in the decoding order. If e.g. only left and top boundaries are known, there is no possibility to know whether an edge leaves the block or not (unless it crosses the left and top boundary only). This ambiguity is shown in **Figure 3.24**. There is no possibility

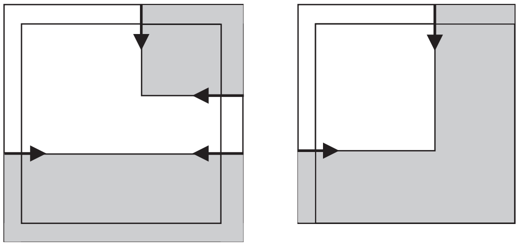


**Figure 3.24**: Distinguishing intersections if all four neighbors (left) and only two of them (right) are known.

to avoid the ambiguity if no information from bottom and right neighbors or from the previous and/or consecutive frames can be used. The performance of spatial error concealment methods in

| Method/case | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| Weighted averaging | 31.71 | 41.52 | 36.86 | 36.03 | 37.63 | 40.08 | 37.52 |
| Maximal smoothness FOD-US | 41.90 | 41.66 | 36.71 | 36.81 | 37.61 | 38.41 | **37.83** |
| Maximal smoothness FOD-EA | 43.46 | 40.81 | 37.40 | **37.16** | 38.28 | 40.42 | 37.53 |
| Maximal smoothness SOD-US | 42.21 | 40.43 | 36.26 | 36.79 | 37.74 | 39.60 | 37.49 |
| Directional interpolation | 42.60 | 30.26 | 27.32 | 22.30 | 21.27 | 41.55 | 35.51 |
| Segmented dir. int. | 42.60 | 30.26 | 38.12 | 28.87 | 23.31 | 43.28 | 34.87 |
| POCS, WA, 5 it. | 42.85 | 33.28 | 28.33 | 21.25 | 23.39 | 40.86 | 35.95 |
| POCS, dir. int. initial, 5 it. | 42.86 | 31.12 | **38.39** | 20.30 | 22.41 | 40.92 | 35.03 |
| Copy-paste | 40.41 | 38.17 | 36.93 | 32.67 | 39.09 | 53.62 | 30.86 |
| MV interpolation | 43.56 | 43.94 | 35.58 | 28.16 | 44.32 | 48.06 | 30.12 |
| MV interpolation SM | 43.92 | 44.72 | 35.77 | 29.82 | 47.74 | 57.46 | 30.75 |
| Boundary matching | 43.92 | 44.28 | 37.02 | 33.21 | 47.86 | 57.46 | 30.68 |
| Block matching | 44.14 | **47.65** | 36.86 | 33.21 | **48.39** | 57.46 | 30.75 |
| Model based (PCA) | **45.24** | 41.11 | 37.72 | 25.53 | 43.21 | **59.88** | 31.41 |

**Table 3.7**: Quality of reconstruction in terms of PSNR (in dB) for the cases shown in **Figure 3.23**. The PSNR is calculated over the $3 \times 3$ MB region with missing macroblock in the middle.The highest PSNR value for each case is marked bold.

scenarios with less then four correctly received blocks surrounding the missing block therefore degrades rapidly. The reconstruction process and thus, the performance of the copy-paste method, decoding without residuals, and temporal MV interpolation are independent of the number of correctly received neighbors.

**Figure 3.25** illustrates part of an I frame from the "panorama" sequence decoded by H.264/AVC JM and concealed *during* decoding i.e. always using only the top and left neighbours, using the concealed macroblock for the decoding and concealment of the following macroblocks. The spatial domain interpolation shows the worst performance. The directional interpolation method should not be used if only two neighbors are available. The "panorama" sequence had a frame rate decimated by four (as usual for mobile communications), resulting in a higher motion. Thus, the copy-paste method does not perform well. Perceptually it causes freezing of the concealed picture parts resulting in overall jerkiness of the video. Better perceptual results were obtained using boundary matching, but having a closer look there are still some small artifacts left. The block matching algorithm leads to the best results. Perceptually, $8 \times 8$ block matching performs already sufficiently.

### 3.3.3 Computational Complexity

Computational complexity is difficult to compare since it depends on the implementation. In this section, the number of operations and size of the memory will be estimated for the presented error concealment methods.

**Table 3.8** summarizes the estimated number of additions and multiplications explained below. It is assumed that a block of size $M \times N$ luminance has to be reconstructed using four neighboring

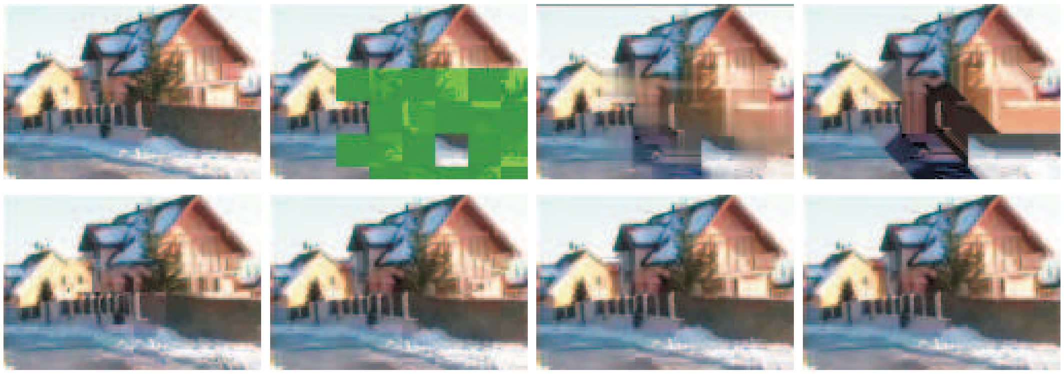


**Figure 3.25**: Screenshots of a part of an I frame in the "panorama" sequence: compressed original (Y-PSNR= 35.86 dB), error pattern (Y-PSNR= 10.45 dB), weighted averaging (Y-PSNR= 18.09 dB), directional interpolation (Y-PSNR= 16.57 dB), copy-paste (Y-PSNR= 22.76 dB), boundary matching (Y-PSNR= 26.27 dB), 8 × 8 block matching with (Y-PSNR= 30.27 dB), 2 × 2 block matching with (Y-PSNR= 30.74 dB).

| method/number od neighbours | complexity |
|---|---|
| weighted averaging | $\mathcal{O}(N \cdot M)$ |
| directional interpolation | $\mathcal{O}(N \cdot M)$ |
| max. smooth recovery | $\mathcal{O}(N^3 \cdot M^3)$ |
| POCS | $\mathcal{O}(i \cdot N \cdot M \cdot \log_2(N \cdot M))$ |
| spatial MV interpolation | const. |
| boundary matching | $\mathcal{O}(s_1 \cdot s_2 \cdot [M+N])$ |
| block matching | $\mathcal{O}(s_1 \cdot s_2 \cdot b_1 \cdot b_2)$ |

**Table 3.8**: Approximate number of operations for implementing the selected individual error concealment methods.

blocks and/or one neighboring frame.

Averaging requires per pixel three additions and one scaling (multiplication). Weighted averaging can calculate the weights in the runtime or pre-calculate them once and store them in a matrix. For runtime calculation, weighted averaging requires additionally to the averaging four multiplications per pixel. Directional interpolation requires first the edge detection. Calculation of Sobel mask per pixel requires seven additions. It is performed for $2(M+N)$ boundary pixels twice (vertical and horizontal filter), resulting in $28(M+N)$ additions. The choice of the resulting direction consists of calculating the slope (one multiplication) and magnitude of the edges (two multiplications and one addition if square of magnitude is used) and to choose the main direction. The reconstruction corresponds to the weighted averaging out of two pixels only, i.e., it requires $M \cdot N$ additions and $3 \cdot M \cdot N$ multiplications. Thus, the determination of the main direction and interpolation take at least $M \cdot N + 56(M+N)$ additions and $3 \cdot M \cdot N + 6(M+N)$ multiplications.

Both the first and the second order derivative based maximally smooth recovery require storing of the matrix $\underline{\underline{S}}^{-1}$ and $\underline{\underline{A}}^{-1}$, respectively, with $(M \cdot N)^2$ entries and boundary of $2(M+N)$ pixels.

The reconstruction requires one matrix multiplication consisting of $(M \cdot N)^3 - (M \cdot N)^2$ additions and $(M \cdot N)^3$ multiplications. Note that both matrices $\underline{\underline{S}}^{-1}$ and $\underline{\underline{A}}^{-1}$ are in fact very sparse (contains many zero-valued elements) and therefore the actual number of operations will be lower.

Error concealment by POCS is an iterative methods. The number $i$ of iteration will thus also parametrize the complexity. In each iteration, a transformation, filter selection and inverse transformation has to be performed. A fast Fourier transformation (and its inverse) of a $3 \cdot M \times 3 \cdot N$ block has a complexity of $\mathcal{O}(\cdot M \cdot N \cdot \log(\cdot M \cdot N))$ operations. The filtering consists of a multiplication of the transform coefficients by a filter window, thus its complexity is proportional to $N \cdot M$.

All temporal error concealment methods require storing of the whole previous frame(s). There are obviously no operations necessary for the zero motion reconstruction method (copy-paste). MV interpolation averages over MVs of neighboring MBs. For four neighboring MBs, two multiplications and six additions are necessary (for both MV components).

The complexity of boundary matching as well as block matching is additionally parametrized by the size of the search region $s_1 \times s_2$. The similarity metric (e.g. SAD) is calculated $s_1 \cdot s_2$ times. One SAD calculation needs $4(M + N) - 1$ additions for boundary matching. For block matching, the complexity of SAD calculation depends on the size $b_1 \times b_2$ of the blocks used for matching — $4 \cdot (2 \cdot b_1 \cdot b_2 - 1)$ additions are necessary for matching of four neighboring blocks. There are no multiplications necessary.

Model based error concealment is the most complex technique investigated in this thesis. It requires update of the principal components in every step, even if no error occurred. Therefore, its complexity is difficult to compare with the complexity of methods presented above.

## 3.4 Adaptive Error Concealment Mechanism

As shown in the previous sections, the optimal choice of the best performing error concealment is content dependent. Up to now, there have been several proposals on how to choose the error concealment method appropriately — how to combine spatial and temporal error concealment.

In [107] a method is proposed that applies temporal error concealment whenever possible; otherwise it employs weighted averaging. The temporal error concealment technique exploits information from a number of past frames in order to estimate the motion vectors of the lost frame. The motion vectors are used to project the last frame onto an estimate of the missing (parts of) frame. The holes resulting from the possibly overlapping projection are concealed by a spatial error concealment (median of the surrounding pixels). The advantage of this method is its suitability for concealing the loss of entire frames. However, concealing in scene changes and videos with reduced frame rate may result in severe impairments.

In [108] the benefit of combining different error concealment methods for different contexts has been shown. The appropriate method is chosen by means of a classification tree. Eight individual methods are investigated. For I frames, the following methods were considered as applicable: averaging of transformation coefficients, weighted averaging in pixel domain, and copy-paste. If panning was detected, the MV was estimated from previous frames and used for motion compensated reconstruction. For P and B frames, again pixel domain weighted averaging is considered and the motion compensated reconstruction for panning. Additionally, four MV interpolation methods using different combinations of top and bottom MVs were used. The performance of the eight chosen methods, however, is limited (cf. Section 3.3).

The error concealment mechanism provided in the JM H.264/AVC reference software (in version 11.0, the most recent version at the time of writing this thesis) still does not work properly[2]. The implementation reveals two generic methods for the slice concealment: pixel domain weighted averaging for the I frames and side match distortion minimizing MV interpolation for the P and B frames. The error concealment is performed *after* decoding all the correctly received slices in a frame. This allows for using not only the top and left boundaries, but also the bottom and right boundaries (if available), since it does not conceal the macroblocks in the scanning order, but rather column-wise. This method is described in more details also in [109].

The mechanism proposed in [111] improves the JM error concealment so that it allows for temporal error concealment for inter-predicted frames and spatial error concealment for intra-predicted frames. The decision about the usage of temporal and spatial error concealment is based on a scene change detector. The scene change detector relies on a macroblock type matching parameter. If a scene change is detected, spatial error concealment is employed. Otherwise, the motion activity is detected. If it is low, copy-paste error concealment is used. Otherwise, for P/B frames, spatial MV interpolation is applied and for I frames the MV of the co-located macroblock in the previous frame is taken.

Some more refinements to H.264/AVC spatial and temporal error concealment were proposed in [49]. There, spatial error concealment was modified to support one main direction rather than to perform weighted averaging only. The main direction is found using the edge detection algorithm described in Section 3.1.2.1 with 16 possible directions. Temporal error concealment is proposed to be applied to the $8 \times 8$ large submacroblocks rather than to the $16 \times 16$ macroblocks. The decision whether to apply temporal or spatial error concealment is based on a scene change detector using the number of inserted I macroblocks.

In this thesis, an adaptive error concealment mechanism is proposed as an alternative to the JM error concealment mechanism and its refinements. The proposed mechanism is based on a decision tree that considers more criteria and better performing error concealment methods than the above mentioned work. A novel scene change detector, independent of the encoder settings and implementation (unlike mentioned state-of-the-art methods) is designed and evaluated as one of the means for the selection of the appropriate method. According to the performance evaluation presented in Section 3.3, the best performing methods for given scenarios are then chosen and the decision tree is designed accordingly. The performance evaluation of the mechanism concludes this chapter.

### 3.4.1 Scene Change Detection

A scene is a sequence of frames between two scene changes. The most abrupt of scene changes are *scene cuts*. A scene cut occurs if the $n$th frame belongs to one scene and the $(n+1)$th frame to an other scene. Additionally to scene cuts, there are *gradual scene changes* like transition, dissolving image, zoom-in/out, wipe and others [110]. In this section, the scene cuts and fast gradual changes

---

[2] The decoder crashes and/or leaves green-coloured unconcealed areas if the error concealment functions in the files `erc_do_i.c` and `erc_do_p.c` are used.

are focused on. Note that frame rate reduction may change some fast gradual changes to cuts.

#### 3.4.1.1 Previous Works

In [49] it is proposed to detect the scene changes in H.264/AVC using the number of inserted I MBs. The way of inserting the I MBs is not standardized. Typically, it is to be inserted if the intra residuals are too large — in such case the scene change detector would work properly. However, the I MBs can be inserted also randomly to refresh the stream, which would result in wrong decisions of the scene change detector. Similarly in [111], a scene change detector for H.264/AVC is proposed that relies on the MB type matching parameter. This parameter compares MBs at the same location in the previous and the following frame with respect to the frame where a loss occurred. Only the MB types are compared, which reduces the computational complexity and avoids problems with dependence of the successive frame on the distorted frame. For one macroblock, the MB type matching parameter is one if the types of MB at the same location in the previous and successive frame are identical. It is zero otherwise. This scene change detection method, again, is encoder dependent. It will not work properly if rate control is used, that may change the MB types even for identical MBs in different frames. Refreshing by I MBs can also deteriorate the results. Thus, an algorithm for scene change detection, independent of the encoder implementation is necessary.

Many types of scene change detection schemes have been proposed in literature so far. Most of them are based on a similarity measure between two consecutive frames. When this measure reveals a sufficiently big difference, a scene change is declared. These schemes define typically a *fixed threshold*. If the value of the measure exceeds the threshold, a scene change is detected. However, a fixed threshold value cannot perform well for all videos mainly due to the diversity of their characteristics. The key problem is to obtain an optimal value for such fixed threshold. If it is set too high, there is a high probability that some cuts remain undetected. If it is too low, the detection scheme produces false detections. In real world videos, both cases simultaneously may occur. To solve the threshold selection problem, several approaches have been proposed.

In [112], a double threshold (with high and low value) was proposed to eliminate missed scene changes and dismiss false ones. Although it improved the efficiency, results are not sufficient, especially in real-world videos containing high motion like e.g. sport games. In addition to this method, [113] proposes a function-based lowering of the threshold, after a scene change was used to decay from high to lower threshold. This technique was used to avoid false detections close to a real scene change, assuming that scene changes cannot occur immediately after each other. However, in most of these methods an optimal threshold (or two thresholds) had to be determined for each video *in advance*. Other methods were proposed to find automatically an optimal static threshold e.g. using histogram differences [114] and entropy [115], but they still have the disadvantage of a static threshold and therefore they are not suitable for real-time applications. A truly dynamic threshold is presented in [116], where the input data is filtered by a median filter and then a threshold is set using the filtered output and the standard deviation. However, it is not suitable for real-time applications, since the median filter uses the future frames as well. A different approach for variable bit rate video is presented in [117], where the number of bits necessary to encode each inter frame determines the amount of movement. This, however, relays again on the encoder implementation.

#### 3.4.1.2 Dynamic Threshold Technique

In this thesis a dynamic threshold technique presented in [71] is used. In this method a normalized SAD $x_i$ is calculated for the $i$th $M \times N$ large frame as

$$x_n = \frac{1}{N \cdot M} \sum_{k=1}^{N} \sum_{l=1}^{M} |\underline{\underline{F}}_i(k,l) - \underline{\underline{F}}_{i-1}(k,l)|, \tag{3.54}$$

excluding the missing parts (if any). For the proposed thresholding function, local statistical properties of the SAD sequence are used, limited to a sliding window of length $W$. If the actual frame number is $n$, then the window contains the frames from $n - (W+1)$ until $n - 1$. The empirical (sample) mean value $m_n$ and the standard deviation $\sigma_n$ of $X_i$ in the defined window are then calculated as

$$m_n = \frac{1}{W+1} \sum_{i=n-W-1}^{n-1} x_i, \qquad \sigma_n = \sqrt{\frac{1}{W} \sum_{i=n-W-1}^{n-1} (x_i - m_n)^2}. \tag{3.55}$$

Together with $x_{n-1}$ the empirical moments are used to define the threshold $T(n)$ as

$$T(n) = a \cdot x_{n-1} + b \cdot m_n + c \cdot \sigma_n, \tag{3.56}$$

where $a, b$, and $c$ are predefined constants. Alternatively, (3.56) can be interpreted as $T(n) = a \cdot (X_{n-1} - m_n) + (b + a) \cdot m_n + c \cdot \sigma_n$, which is more illustrative for the choice of appropriate constant values. The constants $a, b$, and $c$ are very important because they determine the character of the thresholding function. If $b$ takes on high values, the threshold will become more rigid, keeping high values without approaching the $x_i$ sequence. This avoids wrong change detections like in case of intense motion scenes. On the other hand, the detector can miss some low-valued, difficult-to-detect scene changes. A low value of $b$ allows for detecting the regions with high activity, which can provide valuable information to other applications (e.g. bit-rate control, semantics, error concealment etc.). As $\sigma_n$ is the standard deviation, high values of $c$ prevent from detecting intense motion as a scene change. On the contrary, $a$ must have a small value because, as already discussed, the properties of $x_i$ are not always welcome in scene change detection.

In this thesis, the values $a = -0.8$, $b = 2$, $c = 2.5$, and $W = 20$ were used. These values were chosen empirically after testing various test sequences. An alternative to the empirical approach is optimization of a cost function composed of the desired ratio between false alarms and missed detections. However, this ratio is also difficult to choose. The empirical approach was selected because in many cases it was difficult to decide if a scene change should be classified as a cut or not, which makes the optimization problem even more difficult to configure. The performance of such scene change detector can be seen in **Figure 3.26** compared to the fixed threshold. The figure shows the SAD, dynamic threshold and fixed threshold for the chosen frames from the “soccermatch” video sequence. It can be clearly seen that the fixed threshold (horizontal black line) results in both missed and false detections in the cases where the dynamic threshold adapts and performs the correct decision. In the literature [118], the performance of scene change detectors is typically evaluated by means of *recall* $\rho_r$ and *precision* $\rho_p$ ratios defined as

$$\rho_r = \frac{N_c}{N_c + N_m}, \qquad \rho_p = \frac{N_c}{N_c + N_f}, \tag{3.57}$$

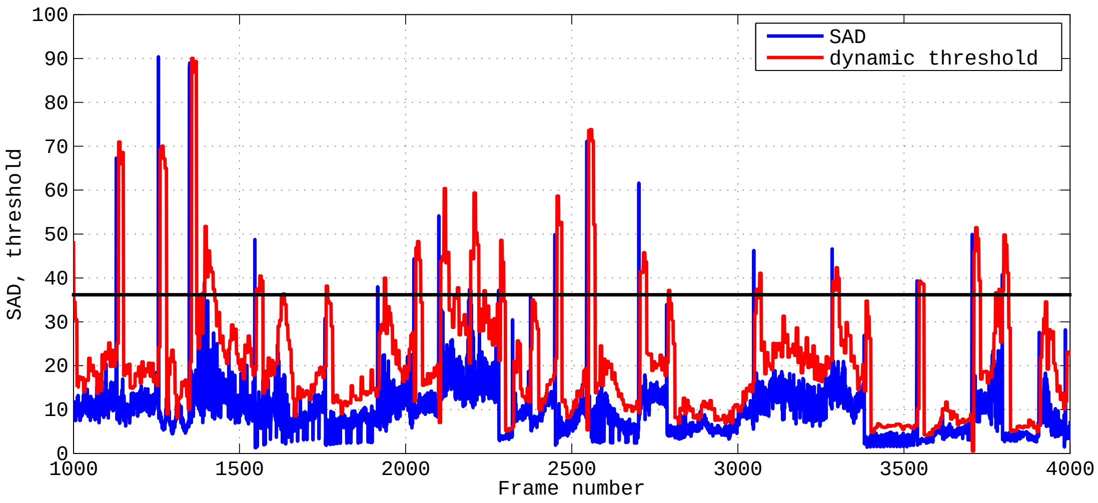


**Figure 3.26**: Performance of the dynamic thresholding technique in comparison with the optimal fixed threshold (horizontal black solid line).

where $N_c$ is the number of correct detections, $N_m$ is the number of missed detections, and $N_f$ is the number of false detections. The tested "soccermatch" video sequence contains 64 scene cuts, 60 of which were correctly detected, four were missed and six were false. This results in recall and precision as listed in **Table 3.9**. The optimal fixed threshold was set after viewing the entire sequence to the value that minimized the number of missed and false detections.

| thresholding | recall $\rho_r$ | precision $\rho_p$ |
|---|---|---|
| optimal fixed | 0.67 | 0.78 |
| dynamic | 0.93 | 0.91 |

**Table 3.9**: Performance of fixed and dynamic thresholding.

Note that sometimes the real positions of the cuts (called *ground truth*) cannot be determined unambiguously since it depends on the amount of change in the scene — e.g., if two cameras are near to each other and there is a cut from one to the next, the picture does not necessarily change much. Moreover, the desired output of the scene change detector depends on the desired application. Especially in the case of fast gradual scene changes that are not cuts (e.g. fast zoom, pan or transition), it is difficult to determine if it "should" be detected or not. The six false detections obtained by the dynamic thresholding techniques were all caused by a very fast transition. These false detections, however, are rather advantageous for the error concealment decision, since the temporal interpolation would not perform well in such cases anyhow.

### 3.4.2 Decision Tree

Clearly, the presence of scene cut is a limiting factor for the temporal error concealment. Thus, the decision about the usage of the temporal and spatial error concealment is based on the scene change detector. If a scene change is detected (`scene change`), spatial error concealment is applied. Otherwise, temporal error concealment is employed. Note that the SAD calculation necessary for the scene change detection according to Section 3.4.1.2 reduces for P/B frames to simply adding up the residuals. Thus, the additional complexity is negligible.

The applicability of the spatial error concealment is also influenced by the spatial characteristics of the neighborhood of the missing part of the picture. Whereas directional interpolation performs well for the cases where all the neighboring boundaries/blocks are known, its application to the cases with only one or two known boundaries may cause unpleasant artifacts. In such situations, weighted averaging usually provides better results (and requires lower computational effort at the same time). Thus for directional interpolation at least three correctly received neighboring macroblocks (`enough neighbors`) should be required. The performance of methods based on directional interpolation is also reduced if there are no clear edges, i.e. if the area to be concealed is surrounded by a multitude of edges with low magnitudes. A decision of `clear edges` is based on the threshold set by the edge detector presented in Section 3.1.2.1.

The efficiency and employability of the temporal error concealment methods depend particularly on the information available. If the concealed frame is an inter-predicted frame and the data partitioning is applied so that the MV of the lost MB is known (`MV correct`), then decoding without residuals provides best results. If no data partitioning is configured, for an inter-predicted MB with known MVs of at least one neighboring MB, the MV interpolation (possibly with side matching) is the best choice. If the concealed macroblocks is within an intra-predicted frame/slice (`I frame`), then block matching algorithm is applied to estimate the MVs.

Following is the pseudo-code summarizing the process of a novel proposed error concealment selection mechanism. If the whole frame is lost, the previous frame is repeated. The first frame in the sequence is always concealed by spatial error concealment (directional interpolation or weighted averaging according to the amount of neighbors available and the detected edges). Directional interpolation with support for more edges is implemented as presented in Section 3.1.2; the maximum number of segments is eight. This method is preferred over the edge aware maximally smooth recovery, since it has lower computational complexity and in most cases performs better for the low resolution pictures. The POCS method performs slightly better than directional interpolation for some scenarios, but it also requires considerable computational effort since it works iteratively. Weighted averaging is chosen instead of the maximally smooth recovery since it performs better in more cases. From the

```
if scene change
   if clear edges AND enough neighbours
      method = directional interpolation
   else
      method = weighted averaging
else
   if I frame
      method = block matching
   else
      if MV correct
         method = decod without residuals
      else
         method = MV interpolation
```

temporal error concealment methods, the subspace methods were excluded due to their complexity and relatively small additional gain. Block matching was based on $4 \times 4$ blocks as well as the motion vector interpolation. In this implementation, the MV interpolation does not support side match enhancement.

### 3.4.3 Performance Evaluation

All chosen methods were implemented into the JM H.264/AVC reference software (see Section 1.2.5) and tested using video sequences described in Appendix B. The sequences were chosen to cover scenarios with different amounts of spatial information, scene change types, motion types and speeds. For all presented experiments the following parameters were set: a fixed number of 500 bytes per slice, I frame inserting after each 15 frames, QP = 28 for both I and P frames, CAVLC. The B frames, rate control, and additional error resilience features were not used. The number of reference frames for motion compensation was limited to five at the encoder; error concealment, however, exploited only one previous frame.

In **Table 3.10** and **3.11** the Y-PSNR of the proposed method and of both the reference and JM method is shown for the original and reduced frame rate, respectively. The reference method is the simplest combination of weighted averaging spatial error concealment for I frames and a copy-paste temporal error concealment for P frames. The JM method corresponds to the error concealment implemented in the JM H.264/AVC reference software that was corrected in order to allow for comparison. The error-free Y-PSNR is the Y-PSNR averaged over all frames of the particular video sequence compressed and decoded without any errors. Frame rate reduction leads to increase of the temporal activity and may result in selection of different error concealment mechanisms. For these experiments a memoryless IP error model (see Appendix D.3.1) was considered with packetloss probability of $p = 7\%$ resulting in the loss of the whole slice, no data partitioning was applied. The average Y-PSNR is calculated over all sequence frames and averaged over 15 different packetloss realizations. To give a hint about the sequence character and the impact of an error at the IP layer, the average size of an I and P slice in macroblocks is presented in the tables, too. The biggest improvement is obtained for the "panorama" sequence, containing linear movement only. The improvement is even higher for the reduced frame rate, as the movement remains still linear and MV estimation can be performed appropriately, whereas the reference method performs poorly since all the panorama details are shifted more, causing lower Y-PSNR. The improvement for other sequences is lower with reduced frame rate, because if there is a non-linear movement, the MV estimation is not sufficient to conceal if the residuals are not known. The smaller the frame rate, the bigger the shift of the moving object from frame to frame. The "squash" sequence shows good performance already for the reference method as only the players are shifted, the background is static. The size of players was small compared to the size of the frame (approximately 10% of the frame width and 40% of the frame height), but not as small as in the "soccer" sequence, thus the proposed method was still able to compensate the movement.

The improvement that can be achieved in the case of a scene cut is limited by the performance of spatial error concealment, which does not perform well if the concealed area is large and the resolution small. Slow transitions are not recognized as scene changes due to our dynamic threshold

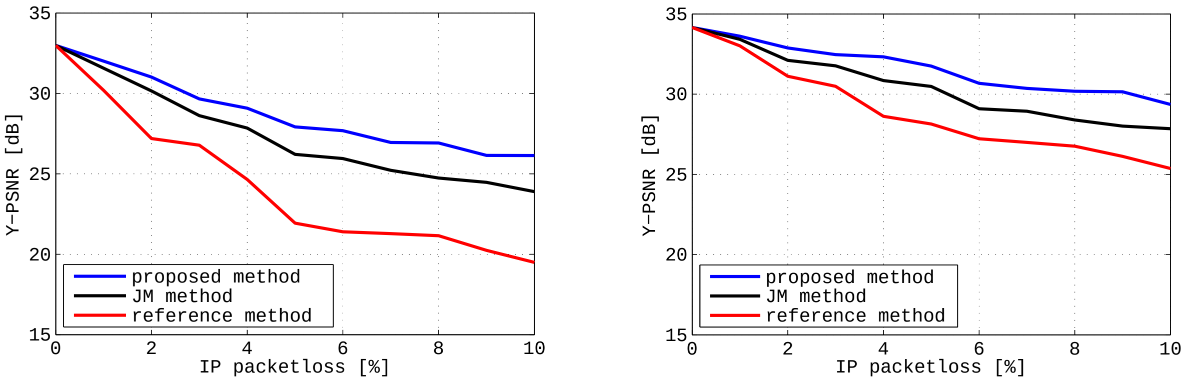
| Sequence name | Average size I/P slice [MB] | Error-free Y-PSNR [dB] | Reference method Y-PSNR [dB] | JM method Y-PSNR [dB] | Proposed method Y-PSNR [dB] |
|---|---|---|---|---|---|
| foreman | 13/60 | 35.24 | 26.63 | 29.21 | 32.55 |
| akiyo | 17/99 | 38.38 | 31.51 | 35.43 | 37.46 |
| mobile | 5/20 | 32.98 | 22.89 | 26.79 | 29.90 |
| panorama | 11/53 | 35.14 | 25.69 | 29.06 | 32.68 |
| soccer | 14/43 | 34.14 | 27.18 | 30.11 | 31.42 |
| squash | 21/79 | 37.56 | 31.82 | 33.87 | 35.34 |
| videoclip | 11/38 | 35.95 | 25.25 | 27.54 | 29.17 |

**Table 3.10**: Results for various test sequences (Y-PSNR averaged over 15 realizations of memoryless IP packetloss), $p = 7\%$, frame rate of 30 f/s.

| Sequence name | Average size I/P slice [MB] | Reference method Y-PSNR [dB] | JM method Y-PSNR [dB] | Proposed method Y-PSNR [dB] |
|---|---|---|---|---|
| foreman | 13/38 | 26.09 | 27.98 | 30.40 |
| akiyo | 17/99 | 31.79 | 34.01 | 36.48 |
| mobile | 5/15 | 21.28 | 25.22 | 26.95 |
| panorama | 11/40 | 22.36 | 27.18 | 30.14 |
| soccer | 14/29 | 26.88 | 28.94 | 30.37 |
| squash | 21/50 | 30.21 | 32.85 | 34.56 |
| videoclip | 11/15 | 24.21 | 25.23 | 27.52 |

**Table 3.11**: Results for various test sequences (Y-PSNR averaged over 15 realizations of memoryless IP packetloss), $p = 7\%$, frame rate decimated by four.

causing the algorithm to use temporal concealment and achieving better performance in such cases. In **Figure 3.27** Y-PSNR over packet loss for the “mobile” and “soccer” sequence is shown. Both video sequences were concealed by the reference method and the proposed method. The Y-PSNR values are averaged over 15 realizations of each particular packet loss. We obtained higher Y-PSNR improvement for the “mobile” sequence, as it contains mostly linear movement that can be compensated well for by the proposed method. In the “soccer” sequence there is non-linear movement (players appearance is changing from frame to frame). Moreover, the weighted averaging error concealment is in most cases sufficient to conceal the playground, hence, the gain by improved spatial error concealment is smaller. The JM method performs obviously better than the reference method, but considerably worse than the proposed method. The main improvement of the proposed method in comparison to both reference and JM method is the application of the temporal error concealment also for the I frames. Improved spatial interpolation also contributes to its better performance, together with the block matching that allows a precise motion vector estimation. The proposed method provides better results than the method presented in [49]. The authors of [49] claim the improvement of in average 1 dB compared to the JM method, while the mechanism proposed in this thesis provides an improvement of in average



(a) ‘mobile’

(b) ‘soccer’

**Figure 3.27**: Y-PSNR over packet loss probability $p$ at the IP layer for the sequence “mobile” and “soccer” with frame rate reduced by four.

2 dB compared to the JM method. Thus, the proposed method seems to perform even better (as the implementation of the method presented in [49] was out of the scope of this thesis, only the information provided in this article can be taken into account when comparing.). Note that all individual error concealment methods used in the proposed mechanism only use simple operations that are performed also in an encoder and decoder (best match search, SAD, edge detection) and thus the complexity remains acceptable for streaming applications.

# Chapter 4

# Cross-Layer Design

## Contents

**4.1 Link Layer Aware Error Detection . . . . . . . . . . . . . . . . . . . . . . 96**
  4.1.1 Error Detection at RLC Layer . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 96
  4.1.2 RLC PDU Based VLC Resynchronization . . . . . . . . . . . . . . . . . . . . . . . 97
    4.1.2.1 Side Information Encoding . . . . . . . . . . . . . . . . . . . . . . . . . 98
  4.1.3 Error Detection and VLC Resynchronization Efficiency . . . . . . . . . . . . . 99
**4.2 Link Error Prediction Based Redundancy Control . . . . . . . . . . . . . 100**
  4.2.1 Link Error Prediction . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 101
  4.2.2 Redundancy Control . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 102
**4.3 Semantic Aware Scheduling . . . . . . . . . . . . . . . . . . . . . . . . . . . . 105**
  4.3.1 Scheduling Mechanism . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 106
  4.3.2 Performance Evaluation . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 108
**4.4 Distortion Aware Scheduling . . . . . . . . . . . . . . . . . . . . . . . . . . . 112**
  4.4.1 Scheduling Mechanism . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 112
  4.4.2 Estimation of Intervals with Low Error Probability . . . . . . . . . . . . . . . 113
  4.4.3 Distortion Estimation . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 114
    4.4.3.1 Data Set Acquisition . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 115
    4.4.3.2 Empirical Distortion Modeling . . . . . . . . . . . . . . . . . . . . . . . 116
  4.4.4 Performance Evaluation . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 118

REAL-TIME video services use the Real-Time Protocol (RTP) to encapsulate the data stream. The header of the RTP is 12 bytes long. Each RTP packet is encapsulated into a User Datagram Protocol (UDP) packet, which adds a header with eight bytes to the RTP packet. UDP [119] is a transport layer protocol that does not provide any retransmission mechanism. It contains eight bytes, one of them being the checksum for error detection. A UDP packet is further encapsulated into an IP packet. The IP header has a size of 20 bytes for IPv4 and a size of 40 bytes for IPv6. The IP packet enters the UMTS network and is further processed as described below.

If the UDP checksum fails, the whole IP packet is typically discarded at the receiver. However, video applications can benefit from having (at least) damaged data rather than discarded by the network. To improve this situation, UDP-Lite [120] was proposed. UDP-Lite provides a checksum with an optional partial coverage. When using this option, a packet is divided into a sensitive part (covered by the checksum) and an insensitive part (not covered by the checksum). Errors in the insensitive part will not cause the packet to be discarded by the transport layer at the receiving end host. When the checksum covers the entire packet, which should be the default, UDP-Lite is semantically identical to UDP. Compared to UDP, the UDP-Lite partial checksum provides extra flexibility for applications that want to define the payload as partially insensitive to bit errors.

In this chapter, possibilities are discussed, how to exploit the link layer information at the application layer and vice versa. Various cross-layer mechanisms are proposed for error detection and scheduling of video packets of the same user in the UMTS. The investigated scheduling methods utilize the predictability of the UMTS radio link. More details on UMTS architecture and protocols are provided in Appendix C. Moreover, semantic and content of the video is taken into account to determine priority of the packets. The scheduling of the information belonging to one stream of one user as discussed here, can be equally considered as a particular form of unequal error protection.

## 4.1 Link Layer Aware Error Detection

THE smallest unit in which an error can be detected without additional mechanisms (assuming UMTS as the underlying system), is the RLC Packet Data Unit (PDU). The size of RLC PDU is typically 320 bits. To enable the usage of RLC CRC information, this information needs to be passed from the RLC layer to the application layer (video codec). The application of UDP-Lite can ensure that damaged packets are passed to the application layer. The passing of the CRC information can be performed as an implementation specific change for the entities that implement the entire protocol stack (e.g. mobile phones). An alternative was proposed in [121], where a new protocol called Complete UDP is developed, enabling forwarding the link CRC information. This protocol, however, has not been standardized yet.

### 4.1.1 Error Detection at RLC Layer

Having the RLC CRC information at the video decoder, the position of the first erroneous RLC PDU within the slice can be specified. All correctly received RLC PDUs before the first erroneous one can be decoded successfully. This adds no computational complexity and there is no rate increase

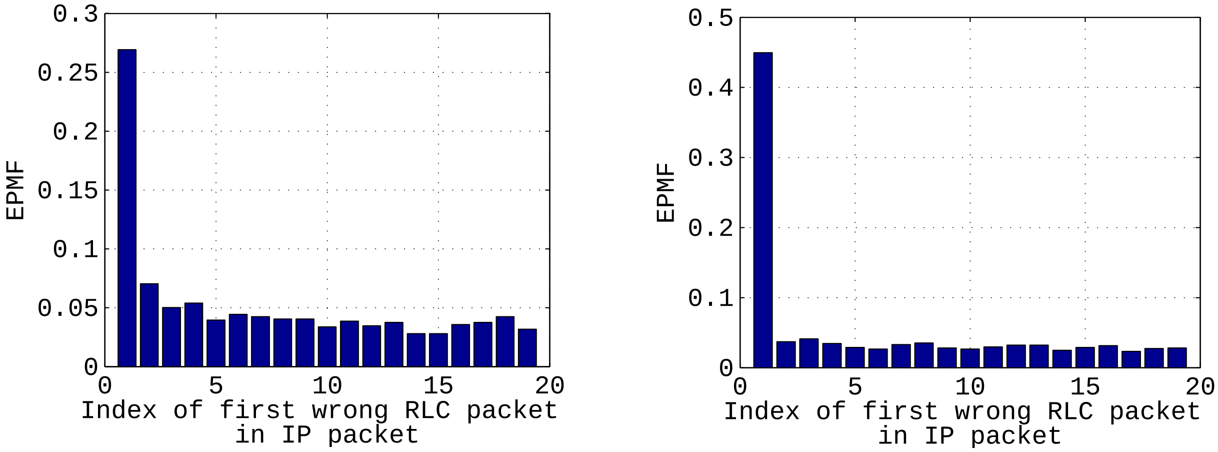


**Figure 4.1**: Distribution of the index of the first lost RLC PDU in a video slice for a static (left) and a dynamic (right) mobility scenario.

necessary. This mechanism is functionally similar to the error detection using a checksum, described in Section 2.4.1. Note that the effect of these two methods on the video quality differs slightly. The checksum based error detection allows for limiting the erroneous picture area since the checksum is calculated over multiples of macroblocks. The RLC CRC is calculated over a fixed number of bits. The size of the corresponding picture area depends strongly on the content and on the compression settings.

The benefit provided by detection at the RLC layer is closely related to the distribution of errors at the UMTS link layer, more specifically to the number of erroneous RLC PDUs within one IP packet. The distribution of the index of the first lost RLC PDU in a slice, illustrated in **Figure 4.1** was obtained by mapping the measured RLC error traces described in Appendix D.2.2 on the IP packets that contain the H.264/AVC encoded “foreman” test sequence. The so mapped RTP video stream was compressed using $QP = 28$; the number of bytes per slice was limited to 700 bytes at most. The peak at the first position (approximately 27% for static and 45% for dynamic scenario) denotes the cases where the detection at the RLC layer does not provide any improvement because the error was detected immediately in the first RLC PDU. In the rest of the cases the method is beneficial at no costs (rate, complexity). Even for the dynamic scenario, this method guarantees improvement in more than 50% of the situations.

### 4.1.2 RLC PDU Based VLC Resynchronization

Due to the desynchronization of the VLC after the first error within the slice, it is not possible to use successive RLC PDUs although some of them might have been received correctly. The start of the next VLC codeword within the segment of the bitstream, corresponding to such RLC PDU payload, is not known. To use all correctly received RLC packets, adaptation of the slice size to the RLC packet size is an option. Such step is indeed connected with rate increase caused by the slice/NAL header and above all by the header of RTP/UDP/IP, which is then equally long as the slice itself (40 bytes if the typical size of 320 bits for RLC PDU is assumed). In Section 2.5 a method of sending the macroblock position indicator every $M$ macroblocks was introduced. This method allowed for resynchronization of the VLC stream at the position of the first VLC codeword that is entirely included within $M$ macroblocks. To utilize all correctly received RLC PDUs, analogically, a side information has to be sent for each RLC PDU to signalize the position of the first VLC codeword in the first macroblock/block that starts inside [73]. If there is no MB starting within the RLC PDU, a special codeword is sent. The isolated macroblocks, separated from the beginning *and* the end of the IP packet by erroneous RLC PDUs, are placed within the picture using a boundary or block matching algorithm as described in Section 3.2.1.3.

The number of cases where this method leads to an improvement is shown in **Figure 4.2**, where the distribution of the number of erroneous RLC PDUs per slice is illustrated for the same video sequence and encoder settings. Only if all the RLC PDUs within an IP packet are erroneous, this method does not succeed. The probability of not succeeding corresponds to the last column of the empirical PMF. A gain is possible in almost 99% of the situations for the static case. It is getting worse for the dynamic scenario, where the probability of all RLC PDUs within an IP packet being erroneous grows to almost 12%. But still, in about 88% of the cases an improvement can be gained by RLC layer detection combined with VLC resynchronization. The peaks in the EPDFs from **Figure 4.2** are caused by the mapping between the TTI and an IP packet; an IP packet can be transported over more TTIs. Moreover, in the dynamic case, a bit stream switching causes variable number of RLC PDUs per TTI. The peak at 8 and 12 thus both correspond mainly to the cases where all RLC PDUs within the TTI were damaged.

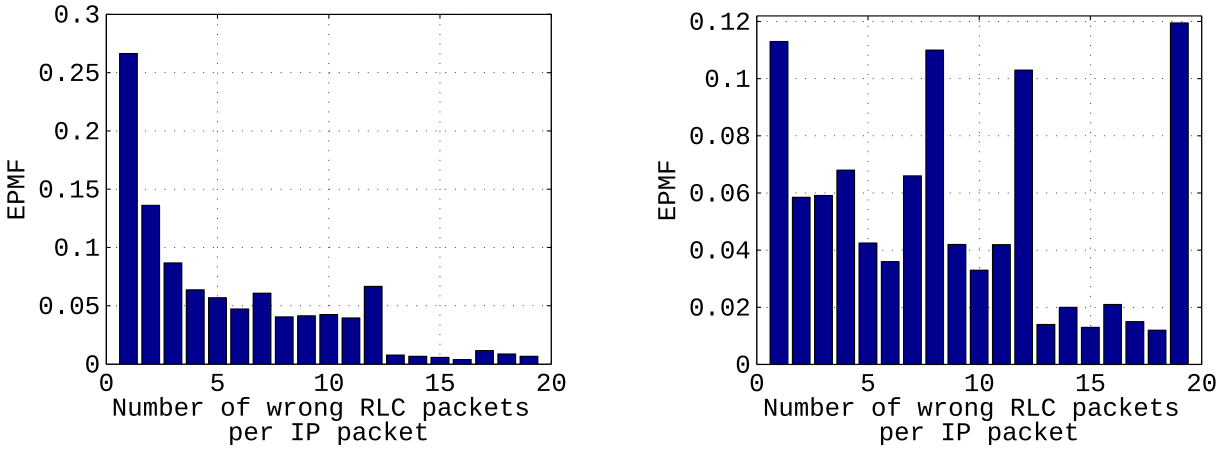


**Figure 4.2**: Distribution of the number of erroneous RLC PDU's per slice for a static (left) and a dynamic (right) UE mobility scenario.

#### 4.1.2.1 Side Information Encoding

The resynchronization information can be advantageously represented as the position of the start of the *first MB starting in the RLC PDU*. The maximum length $l_{\max}$ of such position indicator (PI) within a $q$ bit large RLC PDU is

$$l_{\max} = \lceil \log_2 q \rceil \quad \text{[bit]}, \tag{4.1}$$

| code | $\bar{l}_I$, QP = 20 | $\bar{l}_P$, QP = 20 | $\bar{l}_I$, QP = 30 | $\bar{l}_P$, QP = 30 |
|---|---|---|---|---|
| unary | 78.38 | 90.40 | 96.24 | 43.90 |
| Golomb (ideal) | 7.55 | 7.89 | 8.03 | 6.89 |
| Elias-$\gamma$ | 8.46 | 10.43 | 11.14 | 8.44 |
| Elias-$\delta$ | 7.76 | 9.64 | 10.33 | 8.02 |
| Elias-$\omega$ | 8.39 | 10.54 | 11.28 | 8.91 |
| entropy | 6.07 | 7.50 | 7.80 | 6.61 |

**Table 4.1**: Calculated average codeword length for chosen universal codes, calculated based on the distribution of MBPIs estimated from the "soccermatch" video sequence.

corresponding to fixed-length coding (FLC). For $q = 320$ bits long RLC PDUs the length of the fixed-length PI codeword is $l = l_{\max} = 9$. Fixed length coding has an advantage of robustness compared to variable length coding. The compression gain of a VLC depends on the distribution of the source symbols — here, the source symbols are the values of the PIs.

The Empirical Probability Mass Function (EPMF) of macroblock sizes for I and P macroblocks (for QP = 30 and QP = 20), obtained by encoding the "soccermatch" video sequence is presented in **Figure 2.20**. The distribution of MB sizes corresponds to the distribution of length indicators (LI) in Section 2.5.2. However, it does not correspond to the distribution of the position indicator values. The position of the first MB starting in the RLC PDU is in general uniformly distributed between zero (if an MB begins at the beginning of the RLC PDU) and the size of the previous MB minus one. Nevertheless, the shapes of the LI and PI distributions are similar. Moreover, there are only $q+1$ values to be signalized for a PI — $q$ positions of MBs that begin within one RLC PDU, and one symbol signalizing that there is no MB starting within the $q$ bits of the RLC PDU.

**Table 4.1** presents the mean codeword length of PIs encoded by VLCs introduced in Section 2.5.2. The ideal Golomb code was with parameters $g_{\mathrm{I20}} = 63$, $g_{\mathrm{P20}} = 62$, $g_{\mathrm{I30}} = 63$, and $g_{\mathrm{P30}} = 30$. Note that the higher lossless compression gain for I frame PIs than for P PIs and QP = 20 is caused by the high probability of the I MB size greater than $q$.

The gain from using a VLC code is rather small. The Golomb code with an ideal $g$ is nearest to the entropy, but the gain is not more than 1 bit per codeword. Since an FLC is more robust against errors, it is in this case preferable to encode the PIs.

### 4.1.3 Error Detection and VLC Resynchronization Efficiency

To evaluate the efficiency of the presented methods the JM reference software (cf. Section 1.2.5) was used again. Missing parts of the video were concealed by the copy-paste method. In JM, the RLC segmentation and PI generation were implemented. JM was further adapted to accept link error traces as an input. The baseline profile encoder settings, provided by the configuration file `encoder_baseline.cfg` included in JM, was used with the following changes:

- Slicing with number of bytes per slice limited to 800.
- The period of I frames (GOP size) was set to 50.
- The experiments were performed with $\mathrm{QP} = 20, 22, \ldots, 30$ set equally for both I and P frames.

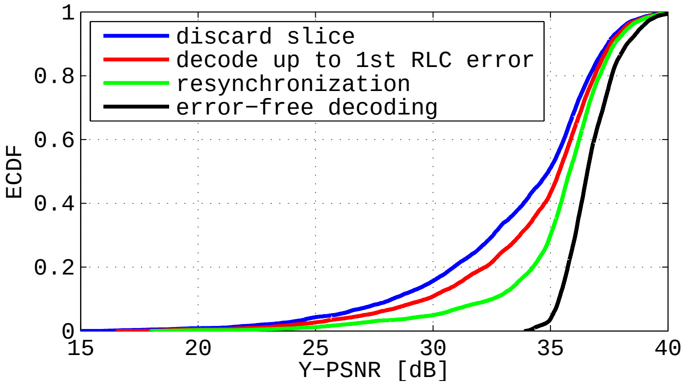


**Figure 4.3**: Quality of video reconstruction at the receiver for different error handling mechanisms.

- The output file mode was set to RTP and RDO as well as rate control were switched off (low-complexity mode).

As a test sequence a “soccermatch” sequence was encoded with a frame rate of 10 f/s, which resulted in a duration of 45 minutes.

**Figure 4.3** shows an Empirical Cumulative Distribution Function (ECDF) of the frame Y-PSNR for different error handling methods for a “soccermatch” video sequence compressed with QP of 26. The two error detection methods (with and without VLC resynchronization, using TB CRC) are compared to the error-free decoding and to the approach where the whole UDP packet is discarded if there was an error detected by UDP checksum. Note that the ECDF contains Y-PSNR of all frames — erroneously received, error-free received but containing impairments due to error propagation, and error-free with compression impairments only. To introduce errors, the RLC error traces obtained from measurements in a live UMTS network (cf. Appendix D.2.2) were used. The overall probability of a TB error was thus 0.266% and the resulting UDP/IP packet error rate was 0.888%. This low error rate is also the reason for the predominantly present high Y-PSNR values. The quality improvement achieved by decoding until the first error occurrence is in 20% of cases greater than 2 dB. The gain achievable by the VLC resynchronization information compared to the decoding until the first error occurrence still is more than 5 dB in Y-PSNR for some erroneous frames, which is significant from the user experienced quality point of view.

On the other hand, the side information introduces redundancy causing a rate increase even in the cases where no errors occur. Therefore, in the next section, a selective scheduling for sending the PI packets is proposed, that only sends them if the link quality sinks.

## 4.2 Link Error Prediction Based Redundancy Control

SINCE the side information for VLC resynchronization is only be employed if an error occurs, it does not make sense to transmit the PI packets if the channel quality is high. However, in wireless mobile networks the channel quality varies very fast. In this section the predictability of the UMTS dedicated channel (DCH) errors is investigated. Moreover, a mechanism is proposed, that transmits the side information only if the predicted error probability exceeds a certain threshold $\gamma$.

### 4.2.1 Link Error Prediction

In wireless systems (e.g. UMTS), transmitter power control (TPC) based on physical link quality is typically used to reduce the effect of fading and to keep the interference level small. In UMTS, the outer loop TPC sets the signal to interference ratio (SIR) target for the inner loop TPC. Within the inner loop TPC, the transmitter adjusts its output power to keep the SIR target according to the feedback commands (power up/down) received [1].

Due to the effects of jointly coded and interleaved TBs within one TTI (Transmission Timing Interval), the UMTS radio link produces error bursts which are separated by short gaps (gap length smaller than or equal to the number of RLC PDUs per TTI). These bursts and short gaps, lying within up to three successive TTIs, then form an error cluster. The power control algorithm of the UMTS DCH separates successive clusters by long gaps in order to reach the link quality target (BLER of 1% in the considered networks) as illustrated in **Figure 4.4**. Such behavior results in correlated TB error statistics at the RLC layer, which results in predictability of errors as shown in [74]. Here, $d_{\mathrm{min}}$ denotes the *minimum delay* for differentiating between the short and long gaps, $d_{\mathrm{FB}}$ is the *feedback delay* necessary to transport the error feedback (FB) from the receiver to the sender, $P_{\mathrm{e}}$ is the probability of error, and $L_i$ is the length of the interval with low error probability.

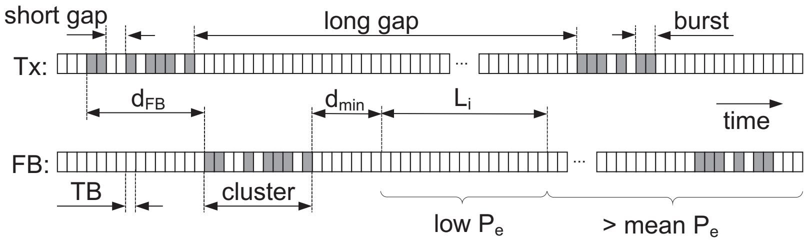


**Figure 4.4**: Schematic illustration of the link error characteristics for UMTS DCH in downlink direction.

Based on these statistics, the Karner UMTS RLC error model (for details see Appendix D.2.2.2) was developed. The TB errors are modeled by a two-state alternating Weibull renewal process. The model describes the length of bursts and gaps. It distinguishes between short gaps occurring in the "bad channel" state of the model and long gaps occurring between the "bad channel" states.

Let an erroneously received TB be indicated by "1" while "0" means an error-free transmission. A positive integer in the exponent determines the number of consecutive erroneous or error-free TBs. For example, the sequence "000001" can be written as "$0^5 1$". A gap with length $m$ (a non-negative integer) is defined as the number of zeros between two ones. Then the expected failure ratio $\widehat{P}(1|10^m)$ (probability of an error after $m$ error-free TBs) for static scenario of UMTS DCH can be estimated as

$$\widehat{P}(1|10^m) = \frac{2m}{a^2}, \tag{4.2}$$

with $a = 3350$ being a parameter of the Weibull distribution. Details to this results are summarized in Appendix D.4. The estimated expected failure ratio $\widehat{P}(1|10^m)$ can be used to predict the probability of the error. In **Figure 4.5** the performance of such error estimator in a UMTS DCH static scenario is depicted. The prediction performance is analyzed for different hard-decision thresholds $\gamma$. If $\widehat{P}(1|10^m) \geq \gamma$ an error is predicted. Otherwise, an error-free transmission is expected.

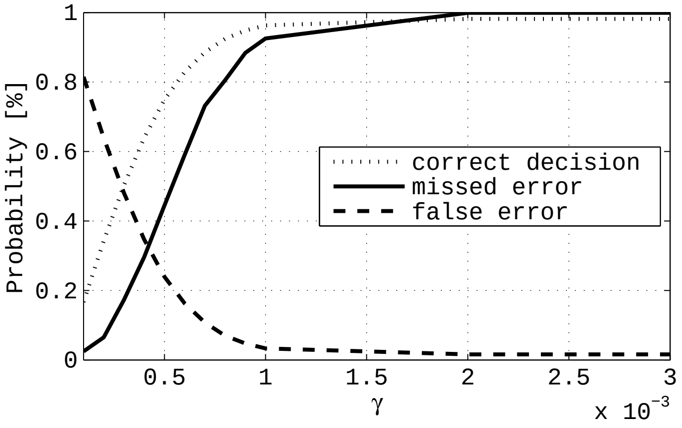


**Figure 4.5**: Prediction performance: Probability of false error, missed error and correct decision dependent on the decision threshold $\gamma$.

The probability of false error $P_\mathrm{f}$ is the conditional probability that $\widehat{P}(1|10^m) \geq \gamma$ given that no error occurred. The probability of a missed error $P_\mathrm{m}$ is the conditional probability that $\widehat{P}(1|10^m) < \gamma$ if an error occurred. The probability of a correct decision $P_\mathrm{c} = 1-[(1-P_\mathrm{e})P_\mathrm{f}+P_\mathrm{e}P_\mathrm{m}]$ is the probability that an error was predicted and occurred plus the probability that an error was not predicted and did not occur; $P_\mathrm{e}$ denotes the overall probability of a link error. To keep the quality high, a low $P_\mathrm{m}$ is needed, whereas to keep the rate low, a low $P_\mathrm{f}$ is desired. Thus, the thresholds around the crossing of $P_\mathrm{m}$ and $P_\mathrm{f}$ will be the most relevant for the investigation.

### 4.2.2 Redundancy Control

As shown in the previous section, a link layer error model can be used for the prediction of errors. Since the PI information is utilized by the decoder only if an error at the radio interface occurs, the rate can be saved by sending the side information packets only if a higher error probability is predicted. **Figure 4.6** illustrates the principle of the redundancy control mechanism if applied to UMTS DCH. At the encoder (located on the application server (AS) in the case of video streaming), the video stream is encoded including the PI side information packets for each video frame and transmitted over the core network to the UMTS Terrestrial Radio Access Network (UTRAN). In UTRAN, scheduling for the radio interface is performed (for more details on UMTS architecture see Appendix C). The information about the current link quality is only available in UTRAN, the AS has no access to such data. Based on the link quality feedback, the error probability is estimated. If the probability of error is low, the side information packets are discarded in UTRAN, whereas if the probability of error is high, the side information packets are sent over the radio interface. Note that the side information can be optionally sent protected by additional encoding, or using acknowledged mode that allows for retransmissions limited by a discard timer. The decision about low/high error probability can be

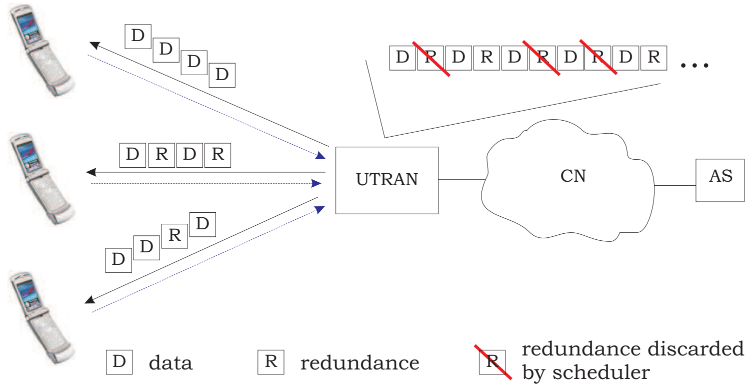


**Figure 4.6**: The principle of the redundancy control performed for UMTS DCH.

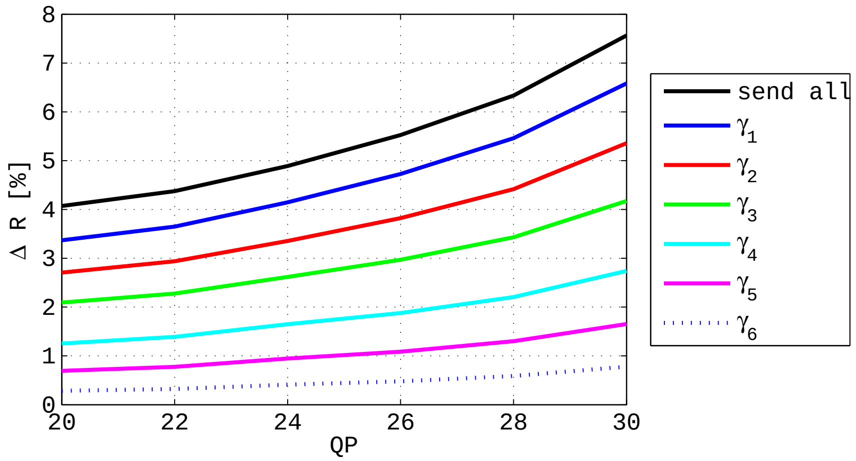


**Figure 4.7**: Rate increase caused by transmitting the side information with different values of the threshold $\gamma_{1-6} = \{1 \cdot 10^{-4}, 2 \cdot 10^{-4}, 3 \cdot 10^{-4}, 5 \cdot 10^{-4}, 7 \cdot 10^{-4}, 1 \cdot 10^{-3}\}$.

based on a single threshold $\gamma$, or alternatively on an adaptive threshold (considering e.g. expected distortion).

For the experiments, again, the "soccermatch" video sequence was used, encoded in the same way as in Section 4.1.3. The same error traces were used for the comparison. To predict the errors the feedback delay $d_{\text{FB}} = 36$ TBs was assumed, corresponding to three TTIs of 10 ms each.

**Figure 4.7** illustrates the rate increase over time for the PIs encoded using a 9-bit FLC and encapsulated into a H.264/AVC NALU and the UDP/IP protocol. The thresholds have the following values: $\gamma_1 = 1 \cdot 10^{-4}$, $\gamma_2 = 2 \cdot 10^{-4}$, $\gamma_3 = 3 \cdot 10^{-4}$, $\gamma_4 = 5 \cdot 10^{-4}$, $\gamma_5 = 7 \cdot 10^{-4}$, and $\gamma_6 = 1 \cdot 10^{-3}$. The quality improvement achievable by VLC resynchronization is illustrated in **Figure 4.8** for all investigated thresholds. The improvement was calculated as a difference between the frame Y-PSNR of the method using resynchronization with threshold $\gamma$ and the frame Y-PSNR of the method decoding up to the first TB error. Note that this ECDF is derived by considering all investigated QPs (unlike **Figure 4.3**). In this representation one can see that e.g. with $\gamma_2$, in more than 7% of

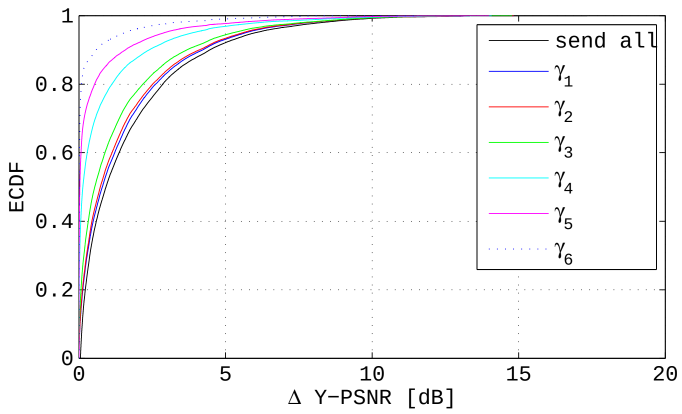


**Figure 4.8**: Quality improvement resulting from resynchronization of VLC, compared to decoding up to the first error occurrence for $\gamma_{1-6} = \{1 \cdot 10^{-4}, 2 \cdot 10^{-4}, 3 \cdot 10^{-4}, 5 \cdot 10^{-4}, 7 \cdot 10^{-4}, 1 \cdot 10^{-3}\}$.

the cases the improvement in quality is higher than 5 dB. Moreover, in 40% of the cases the quality improvement is more than 1.5 dB. The improved frames comprise the frames where an error occurred as well as frames to which the error propagated due to temporal prediction. The improvement for the "send all" strategy is only slightly higher, although it requires more rate.

Of course, looking at the Y-PSNR is not fair without considering the additional rate. Therefore, **Figure 4.9** illustrates the average quality at the decoder normalized by the required rate. As a rate,

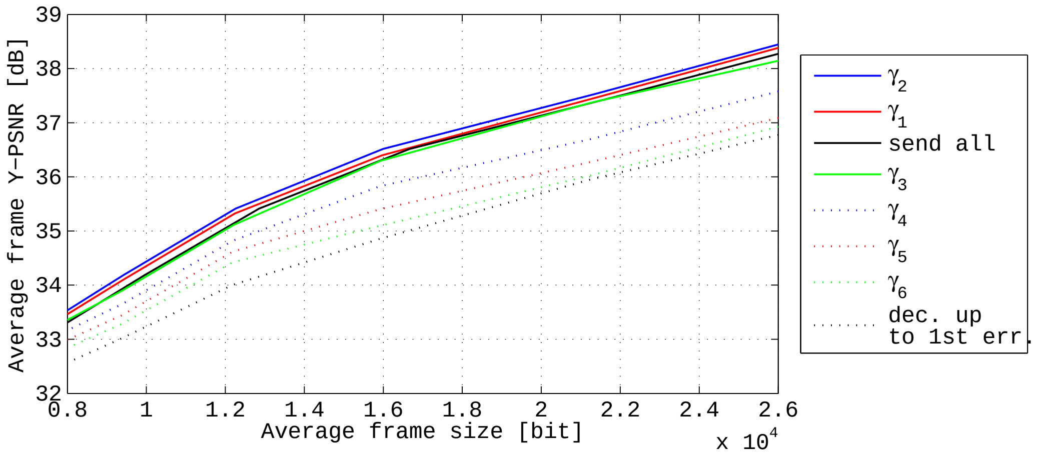


**Figure 4.9**: Average Y-PSNR over the rate. The rate is expressed as average size of frame and side information in bits ($\gamma_{1-6} = \{1 \cdot 10^{-4}, 2 \cdot 10^{-4}, 3 \cdot 10^{-4}, 5 \cdot 10^{-4}, 7 \cdot 10^{-4}, 1 \cdot 10^{-3}\}$).

the average number of bits per frame was taken, that already comprises also the resynchronization information and its header. The averaging was performed over the entire video sequence, containing also the error-free packets. Clearly, the threshold $\gamma_2 = 2 \cdot 10^{-4}$ provides the best quality for the given rate in the whole investigated range. The improvement of approximately 0.25 dB can be achieved by using the error prediction when comparing to the full VLC resynchronization (sending all PIs).

This is a considerable improvement, since the error rate was small and the increase of the rate caused by signalling the side information was taken into account. Furthermore, it can be seen that the resynchronization alone brings approximately 1.5 dB of average quality improvement for the given rate. This also demonstrates, that it is possible to reduce the rate at the radio interface significantly while keeping the same quality at the decoder if VLC resynchronization information is used.

**Figure 4.10** illustrates the Y-PSNR per GOP for the same simulation scenario. This representation provides an example of what quality the user perceives over time (average per GOP corresponding to approximately five seconds of video in our case). The difference between the full resynchronization and resynchronization using prediction (with the $\gamma_2$, that provides the best quality for the given rate) is in most cases zero. Some small differences are caused by the wrong error predictor decision (false detection or missed detection). It can be seen, that resynchronization improves the quality at the receiver considerably, even for a small packet loss rate. The difference to the typically used slice discard method is up to 10 dB per GOP in average, a little smaller gain (up to 8 dB) can be seen compared to the decoding up to the first error. These differences are significant from the user perception point of view. Note that the size of a picture area (in MBs) corresponding to one TB depends on the compression ratio of this area; the compression ratio depends on the content of the video sequence (amount and type of the movement, amount of edges) and on the QP. To limit the erroneous picture area rather than the number of bits, signalization of the LIs at each $M$ macroblocks would be needed as already treated in Section 2.5.

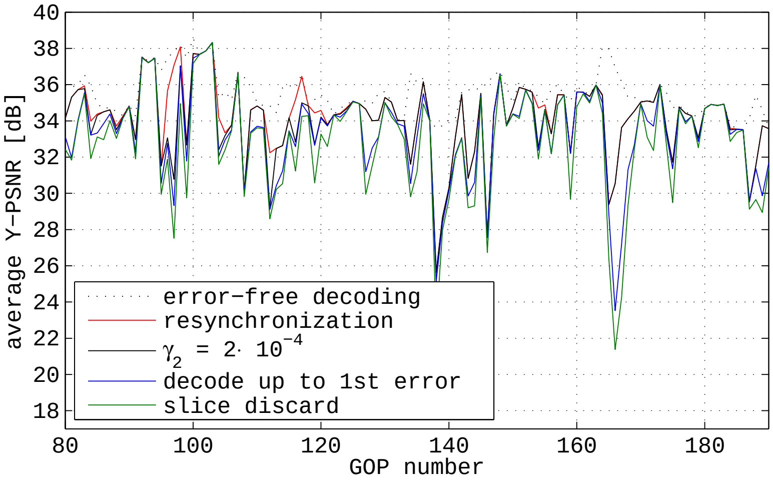


**Figure 4.10**: Average Y-PSNR per GOP over time (GOP number).

## 4.3 Semantic Aware Scheduling

SCHEDULING is a key task of the radio resource management, especially for wireless mobile networks. A lot of publications have already been dedicated to scheduling, some of them handling (wireless) multimedia networks in particular. In [122], a general overview of scheduling algorithms for various services of more users transmitting over wireless multimedia networks is presented. In this section, the scheduling of video packets over UMTS is proposed, the focus is given on the scheduling of video packets from a single stream of a single user.

Already in [123] a truncated power control is introduced for improving the video quality and for efficient transmission of the video stream. In [124] and [125] opportunistic scheduling algorithms are presented which make use of the characteristics of the streamed video data, where in [124] the more important parts of the video stream are transmitted prior to the less important ones in order to ensure more opportunities for retransmissions in case of error. In [125] the priority-based scheduling exploits the diversity gains embedded in the channel variations when having more than one stream. Another approach is shown in [126], where a prediction of the link errors is used in connection with call admission control and scheduling algorithms for avoiding the system of being overloaded and thus improving the quality of the services with higher priority. A semantic-aware link-layer scheduling for video streams of more users over wireless systems was proposed in [26]. It adapts the dropping deadline on the type of the video frame transmitted — the intra frames have the longest delay, inter frames a smaller one. The perceptual importance of the video packets is utilized in [29] for scheduling of retransmissions over 3G networks. An RD-optimized scheduling framework is proposed in [41] with a simple distortion model, in [43] the framework is enhanced.

In this section a scheduling method will be introduced that makes use of the error prediction mechanism presented in Section 4.2.1. Unlike in [123], where transmission is stopped in time intervals where the channel quality is poor, the method presented here makes use of all the available bandwidth but delays the packets with higher importance to a position where smallest error probability is predicted. Unlike [26], the semantically important parts are postponed rather than limiting the dropping deadlines.

### 4.3.1 Scheduling Mechanism

In a video stream, the I frames are more important than the P and B frames, since they refresh the stream. If an error occurs in an I frame, it propagates in the worst case over the whole group of pictures up to the next I frame. Knowing the predicted probability of the error, the transmission of the I frames can be postponed to the intervals where this probability is low [77],[76]. A schematic illustration of the proposed scheduling algorithm is shown in **Figure 4.11** where in the upper part the time series of the received TBs can be seen, with the erroneous TBs marked. The I packets are the RLC SDUs containing parts of I frames. The P packets contain parts of P frames. The general goal of the scheduling algorithm is to transmit the highest priority packets at a time instant where the smallest error probability is predicted. Here, the packets containing I frames are considered more important, having higher priority than packets containing P/B frames. However, another subdivision of priorities would be possible as well. The simple semantic-based two-level priority principle has the advantage of being compatible with the standard video codec settings, i.e. it does not require any data partitioning, other additional error resilience methods, or distortion estimation.

To implement the proposed method in a UMTS scheduler, situated in the RNC at the RLC layer, video semantic awareness is necessary — i.e. the RLC layer must be passed the information about the content of the processed IP packets (RLC layer SDUs). To facilitate this, a cross-layer design must be supported by the mobile network. Passing the semantic information can be achieved in several

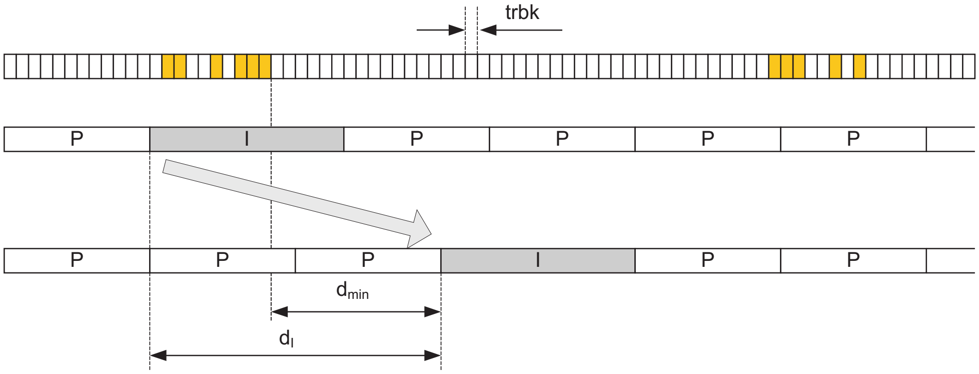


**Figure 4.11**: Schematic illustration of the proposed semantic aware scheduling algorithm.

ways. The first and purely implementation specific way is to read the content of IP packets at the RNC. The possibility of reading the slice headers is not supported to be a possible solution, since it requires RNC to know the particular syntax of a codec. Alternatively, the priority can be signalized either in the NRI field of the video NALUs (cf. Section 1.2.3) or within the IP header as a DiffServ[1] information. Mapping of the IP packets onto UTRAN logical channels according to their priority is also an option, although it requires an increased rate.

The proposed semantic aware scheduling algorithm can be thus applied in UMTS networks without changes to the standard. Additionally to the classical UMTS architecture, at the scheduler (transmitting side) the information about the error status of the TBs at the receiving terminal is necessary. This can be achieved by using the RLC AM mode with the maximum number of retransmissions set to zero. Thus, in UMTS UL all the necessary information for error prediction and scheduling is already available at the mobile terminal.

As the intervals with lower error probability can be predicted after the occurrence of a TB error cluster, the scheduling algorithm maps the I packets onto TBs which are to be transmitted within such a predicted interval. This is performed by delaying the I packets by a time $d_{\text{min}}$ after the last erroneous TB in the cluster (if there is an erroneous TB within $d_{\text{min}}$ the counter for $d_{\text{min}}$ is reset) and transmitting the P packets in the meantime. The delay $d_{\text{min}}$ has to include the maximum of the short gap lengths. The feedback delay $d_{\text{FB}}$ is rather small ($\sim$ 2 TTIs if each TB is acknowledged) in comparison to an average gap length and thus, does not affect the performance of this scheduler essentially. With growing $d_{\text{FB}}$, the effective size of the interval with lower error probability will be reduced from the scheduler point of view. Therefore, possibly lower amount of I packets will be transmitted without any error.

**Figure 4.12** shows the development of the IP packet error probability dependent on the distance to the last TB error cluster. This figure was obtained by simulated transmission IP packets with maximum size of 720 bytes (= 18 TBs), with delays to the last erroneous cluster larger than $d_{\text{min}}$ of 37 TBs ($\geq 3\,\text{TTIs} = 30\,\text{ms}$). At $d_{\text{min}}$ the lowest packet error probability <0.1% is reached, going up to the total mean (over the entire trace) of 0.925% at a distance of 3000 TBs ($\sim$ 2.5 s) after $d_{\text{min}}$. To obtain **Figure 4.12**, simulations have been performed by using a trace generated by the two-layer Karner model (cf. Appendix D.2.2.2).

---

[1] DiffServ (Differentiated Services) is a mechanism for assigning different priority levels to the handling of IP packets [127]. The priority is signalized as a 6-bit value encoded into the differentiated services field of the IP header

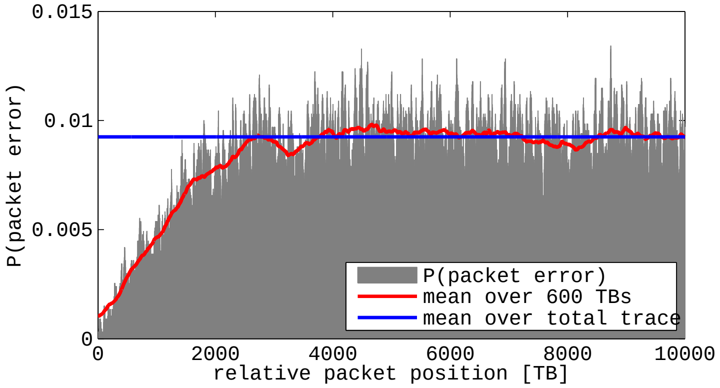


**Figure 4.12**: Packet error probability vs. relative position of a TB to $d_{\text{min}}$ (37 TBs) after the last error cluster.

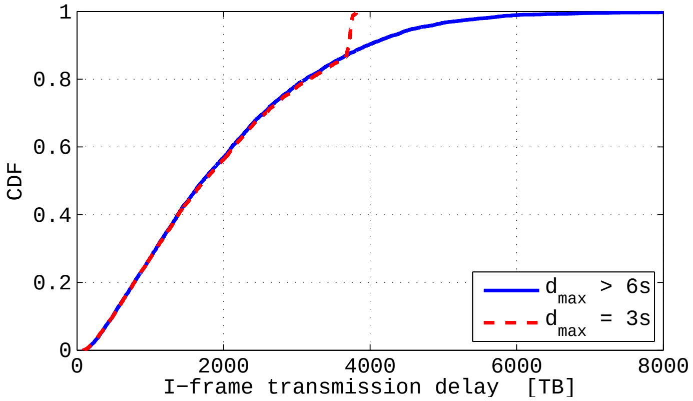


**Figure 4.13**: Resulting I frame transmission delay in number of TBs for the “soccermatch” video sequence.

The proposed method causes an additional transmission delay for the I packets and thus also for the I frames. However, such delay would not cause any deterioration of the quality as long as it remains within the $d_{\text{max}}$, which should be determined according to the application requirements and constraints (play-out buffer). In **Figure 4.13** the resulting I frame transmission delay in number of TBs (12 TB = 10 ms) is shown for a “soccermatch” video streaming sequence, simulated for the measured trace. Without any limitation, the maximum resulting transmission delay for the I frames is about 7000 TBs ($\sim$ 6 s). Thus, $d_{\text{max}} = 6$ s — which is a common value for video streaming services — would be sufficient for a full utilization of the proposed method. Current video streaming applications usually use a 5–20 s large jitter/play-out buffer to cope with the channel fluctuations and the inherent variable-bit-rate nature of coded video sequences [124]. For mobile terminals, the buffer size is smaller.

### 4.3.2 Performance Evaluation

To test the efficiency of the proposed method, the transmission of a video over a UMTS network was simulated. The UMTS network was emulated by means of the error trace obtained from measurements (cf. Appendix D.2.2).

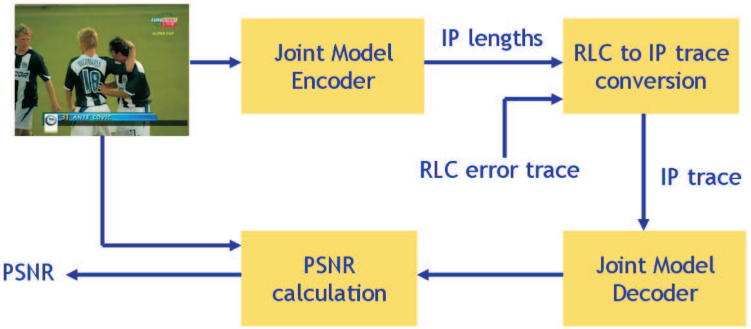


**Figure 4.14**: Block diagram of the scheduling simulation process.

Again, the baseline profile of H.264/AVC is assumed for the experiments, excluding B frames. For these experiments, the JM H.264/AVC encoder and decoder were used as well es a MATLAB program for mapping of the measured RLC trace on the encoded video IP packet trace as shown in **Figure 4.14**. The JM H.264/AVC encoder and decoder were adapted by introducing the following additions:

- The H.264/AVC encoder outputs the IP packet trace. The IP trace captures for each IP packet its size, the type of the encapsulated slice and the error flag is set to zero (no error).
- The H.264/AVC decoder uses an IP packet trace with modified error flags as input. It decodes the error-free slices and conceals the slices corresponding to the erroneous IP packets (error flag set to one).
- The copy-paste error concealment was used for the inter-predicted packets and spatial weighted averaging error concealment was used for the intra-predicted packets.

The simulation starts by encoding the video sequence. The sizes and types of the IP packets together with the IP error trace are fed into the proposed cross-layer scheduler. The scheduler performs the rescheduling of the IP packets based on the probability of the error and constrained by the size of $d_{\max}$. It outputs two IP packet traces with their error flag set to one in the case of an error. The first trace is the one corresponding to the in-order scheduling of IP packets; the second one corresponds to the semantic aware scheduling. The decoding is performed separately for these two traces to enable the comparison.

Two different video sequences with QCIF resolution were chosen: the well-known "foreman" video sequence (400 frames) and the "soccermatch" sequence (11000 frames). The slicing with number of bytes per slice limited to 700 was chosen with a frame rate of 15 f/s. The quantization parameter was set to QP = 25 so that finally a video stream with an average bit rate of in average 300 kbit/s was obtained. No DP and no rate control were used.

To obtain reliable results, the video was decoded several times reusing the whole measured trace for ten times resulting in approximately 10 hours of video streaming. In **Figure 4.15**, the average of Y-PSNR per frame of the "foreman" sequence is shown, corresponding to quality variations seen by the user.

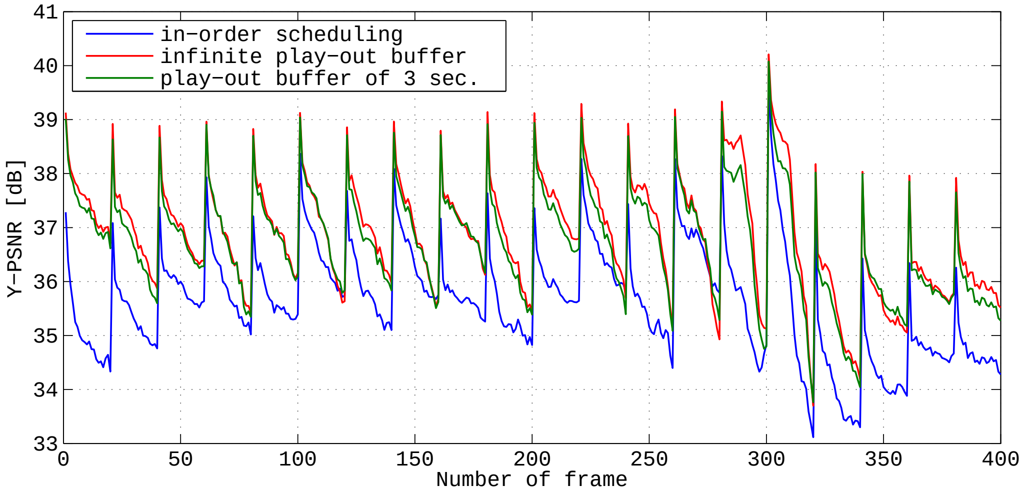


**Figure 4.15**: Average Y-PSNR over the frame number of the “foreman” sequence.

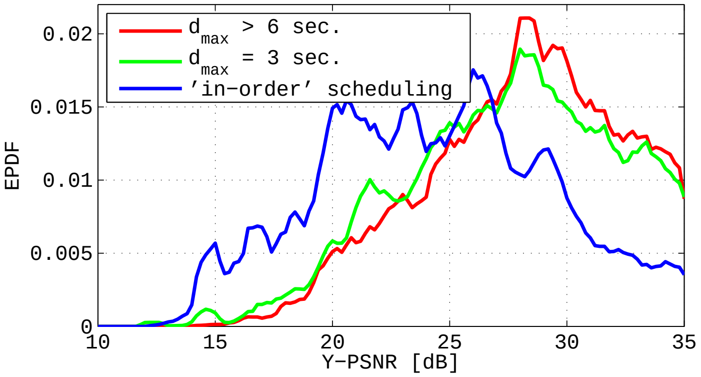


**Figure 4.16**: Empirical PDF of lower Y-PSNRs per frame for the “foreman” sequence.

For the encoding of the “foreman” sequence the I frame frequency was set to 20. Even with the small IP packet loss probability of 0.88%, the average Y-PSNR improvement of more than 1 dB per frame without limiting $d_{\max}$ is obtained. Note that the averaging was also performed over the correct frames. The effect of the proposed cross-layer scheduler on the distribution of the non-averaged frame Y-PSNR is better visualized in **Figure 4.16** for the same “foreman” sequence. The histogram demonstrates that the number of frames with lower Y-PSNRs is reduced whereas the number of frames with higher Y-PSNRs increases. Note that the figure only contains the lower Y-PSNR range. In the higher range, there is a high peak for the Y-PSNRs corresponding to the error-free frames.

The “soccermatch” sequence was encoded with an I frame frequency of 75. The lower I frame frequency better matches to the link error periodicity. Thus, the lower amount of I frames lowers the probability of having an error in such I frame (after scheduling has been applied) due to the fact that efficient scheduling of all packets belonging to one I frame can be performed. In **Figure 4.17** the

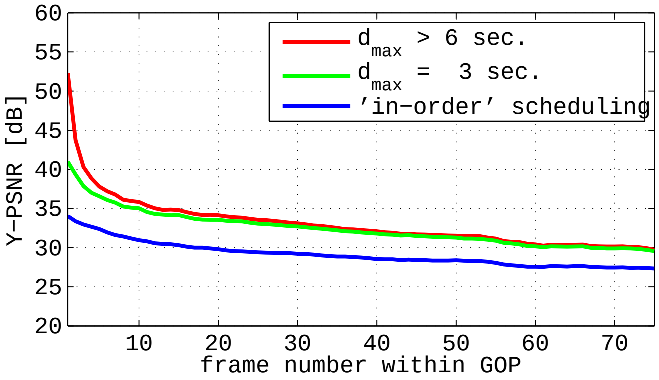


**Figure 4.17**: Error propagation: Average Y-PSNR over the 75 frames long GOP of the "soccermatch" sequence. (The averaging was only performed over the erroneous GOPs.)

average Y-PSNR over the GOP is depicted for the erroneous GOPs only. The error propagation is considerably reduced already for the semantic aware scheduling with $d_{\text{max}} = 3\,\text{s}$. The Y-PSNR results would improve even more if the encoder was set to insert an I frame at every scene cut. The temporal error concealment in the P frames and scene cut situations cause visible artifacts. Another possibility is to assign a higher priority also to the large P frames as it is probable that they contain a new scene.

To demonstrate the benefit of the proposed scheduling algorithm without assuming a particular error concealment method, the frame error improvement $\Delta D$ is defined as

$$\Delta D = \frac{N_{\text{err}} - N_{\text{err}}^{(\text{new})}}{N_{\text{err}}} \cdot 100 \quad [\%], \tag{4.3}$$

where $N_{\text{err}}^{(\text{new})}$ is the number of erroneous frames after applying the improved scheduling and $N_{\text{err}}$ is the number of erroneous frames for the case of "in-order" scheduling. Here, as an erroneous frame, each frame with PSNR differing from the PSNR of the compressed sequence is considered. In **Table 4.4**, the frame error improvement can be seen separately for I and P frames, for both the "foreman" and the "soccermatch" sequences.

| Video sequence | Scheduling method | $\Delta D_{\text{I}}$ [%] | $\Delta D_{\text{P}}$ [%] |
|---|---|---|---|
| "foreman" | $d_{\text{max}} \geq 6\,\text{s}$ | 75.8 | 3.90 |
| "foreman" | $d_{\text{max}} = 3\,\text{s}$ | 39.5 | 2.43 |
| "soccermatch" | $d_{\text{max}} \geq 6\,\text{s}$ | 83.4 | 4.20 |
| "soccermatch" | $d_{\text{max}} = 5\,\text{s}$ | 81.8 | 4.09 |
| "soccermatch" | $d_{\text{max}} = 3\,\text{s}$ | 54.0 | 2.62 |

**Table 4.2**: Frame error improvement for the "foreman" and the "soccermatch" video sequences and different $d_{\text{max}}$ settings.

The difference between the "foreman" and the "soccermatch" sequence is caused by the lower key frame rate in the "soccermatch" case which allows for a more efficient scheduling. In case of a $d_{\max} = 3$ s, the I packets exceeding the maximum transmission delay have to be transmitted immediately and thus cannot utilize the predicted low-error-probability intervals — they experience the total mean error probability.

## 4.4 Distortion Aware Scheduling

A NATURAL enhancement of the semantic aware scheduling presented in the previous section is a refinement allowing to distinguish finer between more important and less important parts of the video. The importance of packet containing a part of the video stream corresponds to the distortion caused by its loss. Thus, he finer the distortion can be modeled, the more efficient will be the scheduling. In this section, a novel distortion estimation mechanism is proposed and evaluated. This mechanism is further used to refine the scheduling method proposed in the previous section as described in the following.

### 4.4.1 Scheduling Mechanism

The proposed scheduling mechanism that makes use of both the link error prediction and the estimated distortion is illustrated in **Figure 4.18**.

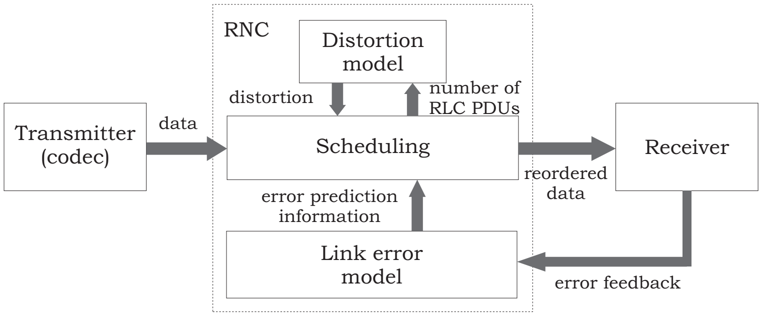


**Figure 4.18**: Functional scheme of the proposed distortion aware scheduling mechanism.

Based on the RLC feedback from the receiver (available at the transmitter after the feedback delay $d_{\mathrm{FB}}$), the link error model block identifies the transmission intervals with low error probability and delivers this information to the scheduler. The importance of packets is a real-valued number, given by the distortion their loss would cause. In Section 4.3, only two priority levels were considered and therefore, the I packets were simply postponed until the start of the interval with lower error probability and then sent. If the number of postponed I packets was greater than the length of the low-probability interval $L_{\mathrm{i}}$, the rest of I packets suffered the overall mean error probability $P_{\mathrm{e}}$. Thus, the knowledge of the interval length $L_{\mathrm{i}}$ was not important. For the distortion aware scheduling proposed in this section, however, this length has to be estimated. The performance of the scheduler can be controlled by the scheduling buffer length $L_{\mathrm{b}} \geq L_{\mathrm{i}}$ and the maximum acceptable delay $d_{\max}$. The delay $d_{\max}$ depends on the requirements of the service; it should be chosen smaller than the application's play-out buffer length. Moreover, the maximum gap length provides a guiding value for determining the size of $d_{\max}$.

Based on the RLC AM feedback from the receiver, the link error model identifies the transmission intervals with low error probability and delivers this information to the scheduler. The distortion model provides an estimation of the *cumulative distortion* $\widehat{D}_\Sigma$ that would be caused by the loss of a particular RLC PDU. This is calculated for every RLC PDU in the scheduling buffer as

$$\widehat{D}_\Sigma = \sum_{k=n}^{m} \widehat{D}_{k-n}, \tag{4.4}$$

where $n$ denotes the number of the frame where the RLC PDU under consideration is located, $m$ is the number of frames in the GOP, and $\widehat{D}_i$ is the estimated distortion in the $i$th frame.

At each step, if an RLC PDU remained in the scheduling buffer longer than $d_{\max}$, then it is scheduled immediately. Otherwise, at each step within the time interval $L_\mathrm{i}$, the RLC PDU associated with the highest estimated cumulative distortion $\widehat{D}_\Sigma$ is chosen from the scheduling buffer. Outside the interval $L_\mathrm{i}$, the RLC PDU with the lowest $\widehat{D}_\Sigma$ is scheduled. If several RLC PDUs have the same (lowest or highest) value of $\widehat{D}_\Sigma$, the oldest among them is scheduled.

### 4.4.2 Estimation of Intervals with Low Error Probability

For a prediction of future link errors, the knowledge of past transmission errors is necessary at the transmitter. That is accomplished in UMTS via the RLC AM feedback. As already discussed, the transmitter receives the RLC AM error information from the receiver after a certain feedback delay $d_\mathrm{FB}$ which is in the order of three TTIs (= 30 ms for the 384 kbit/s bearer). Thus, error prediction within one error cluster is not possible but the location of the error clusters or in other words the length of the long gaps can be estimated.

The long gaps follow a Weibull distribution with a shape parameter $a > 1$ — the distribution is convex from zero up to the inflection point (mode). This leads to a region with a very small probability of having gaps between the maximum of the short gap lengths (here $\sim$ 12 TBs) and about 300 TBs [74]. Therefore, after an error-free time interval $d_\mathrm{min}$ after the last error being at the beginning of an interval with a very low error probability can be assumed due to the convex distribution of the long gaps.

To determine the length $L_\mathrm{i}$ of the intervals with lower error probability, a theoretic analysis with long gaps separating single error events only is performed. Here, the single event corresponds to an error cluster in practice. In **Figure 4.19** the simulated error probability $P_\mathrm{e}$ at a point $x$ TBs after an error event is presented with Weibull-distributed gap lengths. According to this curve the error probability is very small right after an error event, staying below the total mean error probability until approximately 2500 TBs. In practice, $L_\mathrm{i}$ may be estimated via the intersection of this conditional mean curve with the unconditional mean $\bar{P}_\mathrm{e}$.

To obtain an analytic expression for $L_\mathrm{i}$, the steady state error probability $P_\mathrm{e,ss}$ is calculated first. Let $L$ be the random variable measuring the time (distance) between two error events. Assuming that the error process is stationary, Kac's lemma [128] implies that

$$P_\mathrm{e,ss} = \frac{1}{\mathrm{E}\{L\}}, \tag{4.5}$$

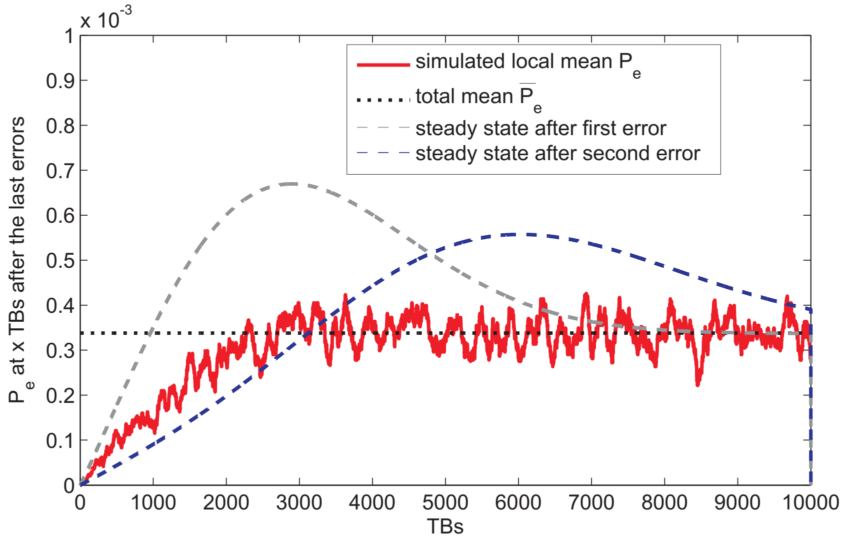


**Figure 4.19**: $P_e$ for Weibull distributed long gaps (Weibull shape parameter $a = 2.176$, scale parameter $b = 3350$).

where

$$\mathrm{E}\{L\} = \sum_{i=1}^{\infty} i \cdot \mathrm{P}\{L = i\} \tag{4.6}$$

is the average recurrence time corresponding to the average gap length (in TBs).

It turns out that the stationarity assumption holds over longer ranges, but not in immediate succession to an error event, where the (measured) error probability is actually lower than the steady state error probability $P_{\mathrm{e,ss}}$. Therefore, the probability $P_{\mathrm{e}}(l)$ of having an error at the $l$th TB after an error event can be approximated assuming that the steady state is reached before the third error after the initial error event:

$$\begin{aligned} P_e(l) \quad &\cong \mathrm{P}\{L_1 = l\} + \mathrm{P}\{L_1 + L_2 = l\} \\ &\quad + P_{e,ss} \cdot \mathrm{P}\{L_1 + L_2 < l\}. \end{aligned} \tag{4.7}$$

Here, $L_1$ and $L_2$ denote the gap lengths before the first and the second error, respectively. The result (blue dashed curve in **Figure 4.19**) shows that taking the intersection between this curve and the total mean as the end of $L_{\mathrm{i}}$ gives a good approximation, whereas the assumption of reaching the steady state after the first error leads to an underestimation of $L_{\mathrm{i}}$ (grey dashed curve).

### 4.4.3 Distortion Estimation

Various distortion models have been proposed in literature so far. In [129] a semi-analytical model is proposed for H.264/AVC assuming the impairments caused by the loss or damage of the frame caused by the decoding of stream with errors (without error concealment). Another approach can be found in [130], where the visibility of packet loss is modeled depending on a set of motion and spatial video parameters. The method proposed in [131] combines the analysis-by-synthesis approach with a model based estimation, which requires decoding of all possible loss patterns to obtain the resulting distortion in the first frame. In [132], the distortion over the whole sequence is estimated for both the compression and the packet loss impairments.

The proposed scheduling mechanism is based on RLC PDUs. Thus, a distortion model is needed with RLC PDU granularity, instead of the more common IP/RTP/NALU granularity. The task of designing such a model is somewhat simplified by the fact that the decoder will not be able to recover information in RLC PDUs after the first lost PDU of a NALU. Thus, if a NALU is made up of $k$ RLC PDUs, only $k$ cases need to be tested. The impairments caused by the loss of an RLC PDU will propagate from the frame of origin to all frames that reference it. Thus, it makes sense to subdivide the distortion estimation into two steps:

1. determine the *primary distortion*, i.e. the distortion within the frame caused by a lost PDU,
2. determine the *distortion propagation*, i.e. the distortion in the following frames until the end of the GOP.

Note that the distortion is estimated after error concealment, and therefore depends on the performance of the latter. The estimation considers the position and amount of lost data, as well as the size of the corresponding picture area. The number of lost/damaged MBs required by the model can be obtained without fully decoding the video as a proportion of the MBs per NALU packet (if it is fixed) corresponding to the proportion of lost RLC PDUs, by reading of the entropy encoded stream (without decoding the video) or by extra signaling.

To investigate the behavior of the distortion with respect to the number of missing bits and macroblocks, experiments were carried out as described in the following.

#### 4.4.3.1 Data Set Acquisition

In order to obtain a data set consisting of measured distortions, several assumptions about channel and encoder settings are necessary. In the considered UMTS architecture, the RLC PDU of size $q = 320$ bits is the smallest data unit allowing CRC error detection. Thus we chose to measure the position and amount of lost data in multiples of $q$. The size of the missing picture area is measured in multiples of macroblocks (MBs). For temporal error concealment, the zero motion method is taken, as it represents the worst case. Spatial error concealment is used only for the first frame in the sequence. The experiments[2] were all performed using the H.264/AVC JM modified to support RLC segmentation. A training set of five QCIF video sequences with various contents and with a frame rate reduced to 7.5 f/s was used to obtain the distortion model: “foreman”, “akiyo”, “videoclip”, “soccer”, and “limbach”. All these sequences were encoded using slices of length limited to 650 bytes, QP = 26, one reference frame only, no RDO, no rate control, no DP, CAVLC entropy coding, and other settings corresponding to the baseline profile. Three more video sequences were used to verify the prediction performance of the proposed distortion model: “silent”, “glasgow”, and “squash”.

The mean square error (1.3) was used as a distortion measure. For each video sequence, each possible position of an RLC PDU error was simulated. There was always only one RLC PDU error per GOP in order to follow the error propagation caused by each error separately. **Figure 4.20** visualizes the data set obtained by these simulations for P frames. The smooth blue region corresponds to the

---

[2] The major part of the experiments for data acquisition was performed within a student’s bachelor thesis [133].

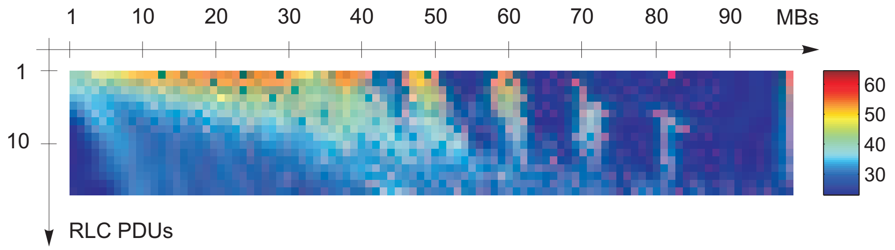


**Figure 4.20**: Primary distortion (expressed as PSNR in dB) averaged over all simulated video sequences, over the number of missing MBs and RLC PDUs.

points where no data were available. The axis origin (number of both missing MBs and RLC PDUs equal to one) is in the top left corner.

The PSNR is not a totally smooth function of the number of MBs and RLC PDUs. In particular, the left part of the plot and the part with more than 50 MBs show several points of high distortion. These correspond to intra coded MBs (I MBs). Unfortunately, this effect on distortion cannot be suppressed or compensated for, since the number of I MBs within the lost part of a slice is not known and their frequency depends on the encoder implementation. Thus, the I MBs inserted into P slices will degrade the performance of the distortion estimation. **Figure 4.21** depicts the the number of I MBs within a slice, again, averaged over all P frames of all considered video sequences. Note, that the

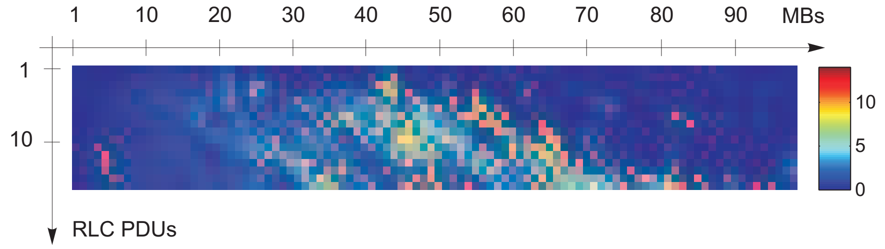


**Figure 4.21**: Image representing the logarithm of primary distortion in I frames averaged over all simulated video sequences.

points with high average number of I MBs really correspond to the points with rarely high distortions in **Figure 4.20**.

**Figure 4.22** illustrates the distortion as a function of the average macroblock size in bits. This representation shows that unfortunately a model reduced only to the average size in bits (ratio between the number of RLC PDUs multiplied by $q$ and the number of macroblocks in the lost part of the slice) is not sufficient to model the distortion. The variance of a distortion for a particular MB size is too high.

#### 4.4.3.2 Empirical Distortion Modeling

In order to build a model based on the obtained data set, the estimated primary distortion $\widehat{D}_0$ has to be represented as a function of the number of missing RLC packets (PDUs) $N$ and the corresponding

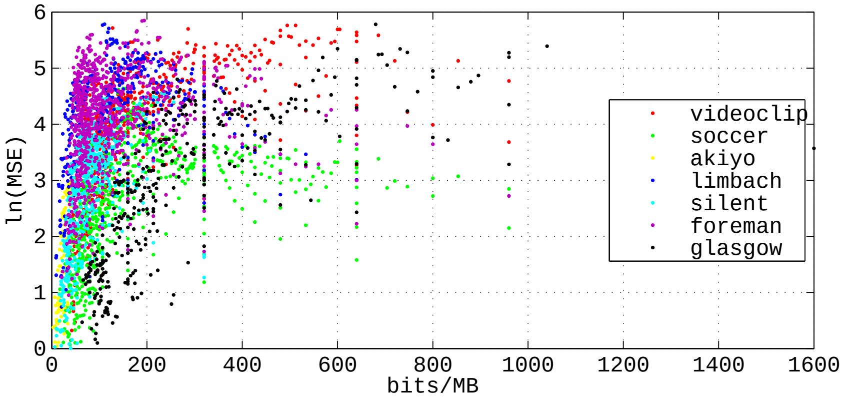


**Figure 4.22**: Dependency of distortion on size of MB in bits.

number of missing macroblocks $M$:

$$\widehat{D}_0 = f(M, N). \tag{4.8}$$

The task is to find an appropriate model function $f(.)$ and its best fitting parameters for the given scenario (i.e. encoder settings/error concealment assumptions).

The distortion grows monotonically with increasing number of missing RLC PDUs. We observed that this growth is better described by a rational rather than exponential function. With increasing number of missing MBs, the distortion decreases.

The following model approximates the measured distortion well as also illustrated in **Figure 4.23**:

$$\widehat{D}_0 = e^{a \cdot M + \frac{b}{N} + c}, \tag{4.9}$$

the parameters $a, b$, and $c$ determined by least square fitting. For the model data set, the following values were obtained: $a = -0.041$, $b = -6.371$, and $c = 4.482$.

In order to model the error propagation in a GOP, all obtained distortion sequences $\{D_k\}_{k=1,2...}$, where $k$ is the frame index, were normalized by the corresponding primary distortion $D_0$. On average, an exponential decrease was observed:

$$\widehat{D}_k = \widehat{D}_0 \cdot e^{-s \cdot k}, \tag{4.10}$$

where $s$ determines the speed of decay, which depends strongly on the nature of the motion in the sequence. Since we assume that only $M$ and $N$ are available, the error propagation has to be estimated by an average over all sequences in the model data set. This degrades the performance of the estimator, since the error propagation differs considerably for different sequences as shown in **Figure 4.24**. The such obtained decay is $s = 0.08$. To obtain a more exact model, more information about the video sequence is necessary. The propagation of errors depends on the movement in the sequence, which also influences the encoding process, e.g. by skipped and intra macroblocks.

The estimation performance of the proposed method was tested for all video sequences in the training set, as well as for the three additional video sequences from the test set. **Table 4.3** summarizes the results in terms of the Pearson correlation coefficient $\rho$ for both primary distortion and error propagation model, obtained by decoding of all possible RLC PDU loss error patterns. The comparison

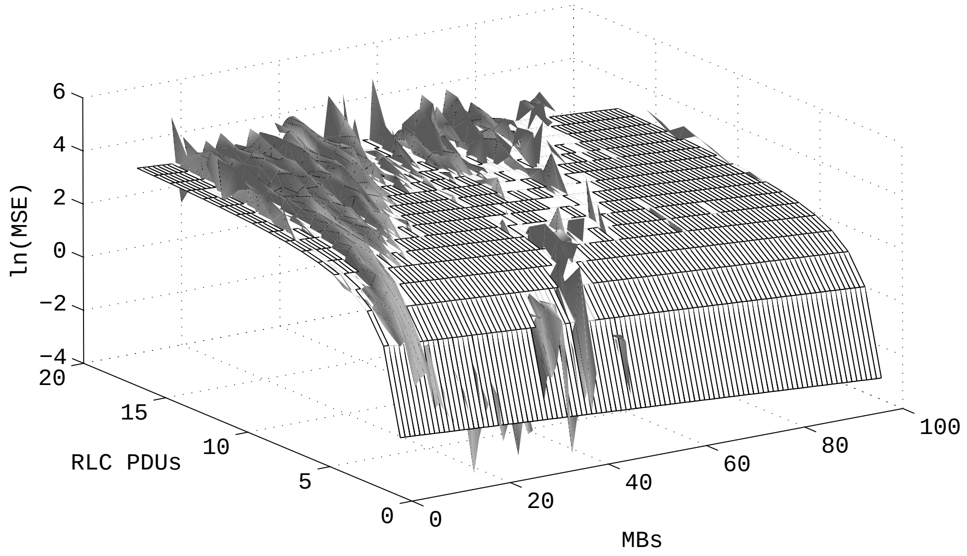


**Figure 4.23**: Fitting of the measured data by the proposed model.

was performed with the estimator based on the whole training set, not per video sequence. The performance of the distortion estimation varies for different video sequences. Especially scene cuts and frames with significant amount of I-MBs cause lower correlation. However, in the scheduling application absolute distortion values are not as important as the relation between the distortions caused by different error patterns.

### 4.4.4 Performance Evaluation

For all experiments, JM H.264/AVC was used. Video sequence "videoclip" containing a music video clip was chosen as a test sequence since it contains a variety of different scenes separated by scene cuts and gradual transitions. The sequence was encoded with the settings corresponding to those used

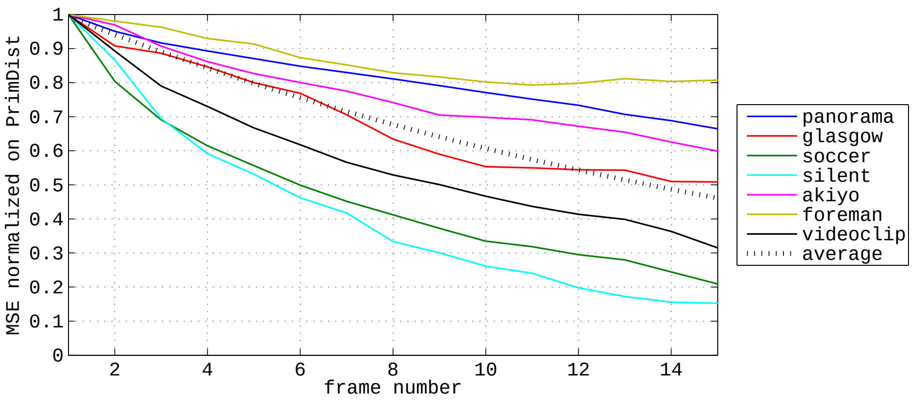


**Figure 4.24**: Error propagation in the frames of the same GOP for various video sequences.

| Video sequence | $\rho(D_0)$ | $\rho(D_k)$ |
|---|---|---|
| "foreman" | 0.86 | 0.84 |
| "soccer" | 0.88 | 0.81 |
| "akiyo" | 0.85 | 0.78 |
| "limbach" | 0.68 | 0.80 |
| "videoclip" | 0.84 | 0.77 |
| "silent" | 0.87 | 0.85 |
| "glasgow" | 0.79 | 0.67 |
| "squash" | 0.83 | 0.73 |

**Table 4.3**: Performance of the distortion estimation for video sequences in the model data set (1–5) and new sequences (6–8).

for obtaining the distortion model. It was decoded 1500 times for each of the considered methods to obtain sufficient statistics. The CRC of the RLC PDUs ($q = 320$ bits) is used to detect the errors and the decoding of a video slice is performed up to the first erroneous RLC PDU within the corresponding packet as proposed in [73]. Following methods are compared:

- **In-order scheduling**: decoding using unchanged link error traces obtained from measurements in a live UMTS network,
- **Scheduling limit**: achievable limit under assumptions that the position of errors is known and both $d_{\max}$ and $L_{\text{b}}$ are larger than one entire video sequence,
- **Semantic aware scheduling**: prioritized scheduling of the I frames as proposed in [77], but assuming the usage of the correctly received data from NALU up to the first missing RLC PDU,
- **Distortion aware scheduling**: configurations of the proposed method for different $d_{\max}$ and $L_{\text{b}}$.

Here, the results of simulations for the following combinations are presented: $d_{\max} = 8400$ TBs, $L_{\text{b}} = 3600$ TBs and $d_{\max} = 3600$ TBs, $L_{\text{b}} = 1200$ TBs (denoted further as 8400/3600 and 3600/1200), which corresponds to the maximum delay of 7 and 3 seconds, respectively.

The empirical CDFs of the frame PSNR for the presented methods are shown in **Figure 4.25**. The difference between the semantic aware scheduling and the distortion aware scheduling method is well illustrated. All CDFs approach one at the PSNR $= \infty$ (error-free frames). The semantic aware scheduling method globally reduces the number of erroneous frames since it moves the errors from the first frame in the GOP to the later ones, making the error propagation chain shorter. The proposed distortion aware scheduling reduces the number of frames with higher distortion and consequently increases the number of frames with lower distortion. It can even happen that the total number of frames containing an error (even small) increases after applying the distortion aware scheduling. This is caused by burst errors possibly affecting multiple frames, since the combinations of errors in time are not considered; i.e. if an error occurs, followed by another error in the next frame at the same position, then the cumulative distortion is smaller or equal to the sum of the two distortions. However,

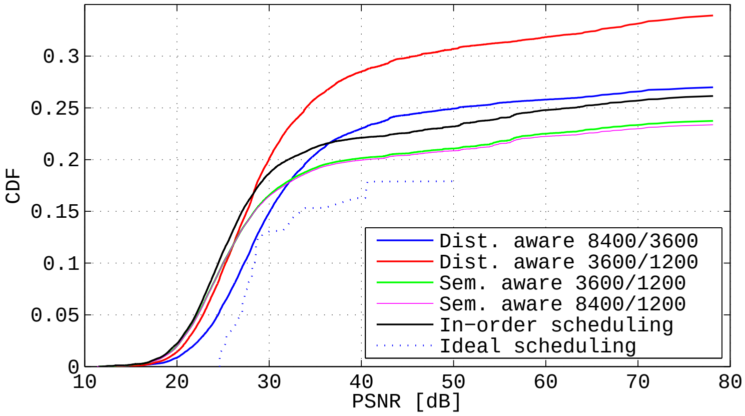


**Figure 4.25**: CDF of the frame PSNR achieved by the discussed scheduling methods.

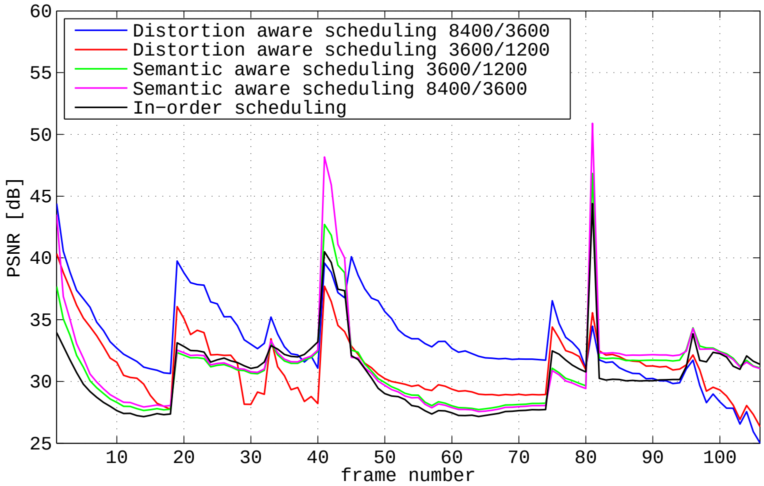


**Figure 4.26**: PSNR in time: comparison of presented scheduling methods.

as there is no information about the motion between the two frames, the additional distortion caused by another error cannot be determined reliably. Moreover, considering all possible combinations of two and more errors would increase the complexity of the scheduler considerably.

**Figure 4.26** depicts the PSNR over the frames of the tested "videoclip" sequence. Again, the difference between the semantic aware scheduling and the distortion aware scheduling can be seen. The PSNR at the end of the sequence decreases for the distortion aware scheduling. This effect is caused by the high predicted distortions for a loss of the first frame. In our simulation we encoded the video sequence 1500 times, thus the first frame follows after the last one again. Since the first frame is concealed only by spatial error concealment, the corresponding distortion caused by a loss of its parts is higher. In **Table 4.4**, the proportion of error-free frames is summarized together with the mean distortion for all presented methods. The semantic aware scheduling is not as sensible to smaller $d_{\min}$ as the distortion aware scheduling.

| Method | Error-free frames [%] | Mean PSNR [dB] |
|---|---|---|
| In-order scheduling | 73.86 | 29.29 |
| Semantic aware scheduling ($d_{\max} = 3600\,\text{TBs}$) | 76.27 | 30.26 |
| Semantic aware scheduling ($d_{\max} = 8400\,\text{TBs}$) | 76.63 | 30.32 |
| Distortion aware scheduling (3600/1200) | 66.07 | 30.44 |
| Distortion aware scheduling (8400/3600) | 73.01 | 32.03 |
| Scheduling limit | 82.09 | 35.53 |

**Table 4.4**: Number of erroneous frames and mean PSNR for the presented scheduling methods.

The reason why semantic aware scheduling performs slightly worse than for the “soccermatch” and “foreman” sequences analyzed before, is mainly the utilization of the “videoclip” sequence containing some inter-encoded scene cuts. Clearly, the quality of the I frames is improved considerably (see frames 1, 40, and 80 in **Figure 4.26**). The only case where the quality after scheduling worsens compared to the quality without scheduling, are the P frames containing a scene cut, e.g. 18, 75. This problem can easily be removed by inserting an I frame in case of a scene cut. Nevertheless, semantic aware scheduling still gains about 1 dB in average.

The proposed distortion aware scheduling method gains in average 2 dB compared to the common (in-order) scheduling. The gain achieved by using the distortion model rather than the semantic information is about 1 dB. These gains are significant from the point of view of user perception.

# Chapter 5

# Conclusions

THIS thesis is dedicated to error resilience techniques for transmission of video streaming over wireless mobile networks. After investigating several possibilities of using the residual redundancy of the video stream at the decoder, it can be concluded that the decoding of the damaged packets indeed provides a powerful means for more robust transmissions, especially if supported by the encoder and/or by a cross-layer design. The recent H.264/AVC video codec is used for all presented experiments. However, most of the proposed techniques may be employed with other video compression standards, too.

One of the proposed encoder-assisted methods is the VLC resynchronization, enabled by sending the out-of-stream side information. In contrast to smaller slice sizes, out-of-stream resynchronization information adds a negligible overhead while providing a considerable quality gain. Moreover, precoding of the out-of-stream side information from more slices/frames enables its transmission via a better protected channel (e.g. a channel allowing for retransmissions). The quality at the receiver can also be improved by implicit detection methods that only require a specific decoder enhancement. Syntax analysis and impairment detection do not violate the video coding standard since they both can be considered as a postprocessing. A remarkable quality improvement can be achieved by these methods in range of bit error probabilities of $10^{-3}$ to $10^{-7}$. However, the detection performance is limited by exploiting of the residual redundancy only. Additional information embedded in the stream whether as watermarking or as an out-of-stream checksum, may greatly help to enhance the reliability of the error detection, while still preserving the low complexity of both the encoder and the decoder. Especially the proposed relation based watermarking scheme provides sufficient scalability that can be employed in a rate-distortion optimized way. Apart from video streaming, these methods can be particularly advantageous for video broadcasting and multicasting applications, which cannot afford any retransmissions, not even at the lower layers. A slightly more complex soft decoding of the VLC fully exploits the residual redundancy in the stream. It can still be considerably improved by combining it with the proposed impairment detection and concealment. A combination with out-of-band signalized resynchronization information brings additional benefits, too.

Even with robust error detection techniques, the loss of packets cannot be avoided completely. If a packet is declared as lost (i.e., either not arrived before expiry of the maximum allowed delay, or arrived damaged), the corresponding missing parts of the video have to be concealed. The proper choice of error concealment mechanism has a considerable impact on the video quality. Due to the real-time constraints, the allowable complexity of postprocessing is usually limited. Temporal concealment works in most of the cases better than spatial concealment, even for intra-predicted frames, where on the other hand it can prevent their refreshing function. Scene changes are a critical point, where spatial concealment has to be deployed. This work includes a method for detection of such scene changes, working in the picture domain, independently of encoder settings. The proposed adaptive error concealment mechanism provided excellent results. Its simplicity minimizes the effort of implementation and it is thus strongly recommendable in practical use.

If the transmission system is known, the exchange of information between different protocol layers can essentially improve error resilience of the video transmission. A very good example of utilizing the UMTS radio link packet information was demonstrated in this thesis. The radio link packets are typically several times smaller than the NALU size and have a CRC information attached. Utilizing this information by the application layer results in neither rate increase, nor additional complexity, while providing a substantial quality improvement. The availability of radio link information at the application layer is therefore undoubtedly recommended. Measurements in live UMTS networks showed that in static scenarios not all of the radio link packets within the distorted NALU are wrong. Thus, the results are even better if combined with the VLC resynchronization in each radio link packet, because then all correctly received radio link packets within a NALU can be used for the error-free decoding. Although the resynchronization side information causes a rate increase, it still provides a benefit in the RD sense, if there are errors at the link layer. It was shown that especially in the static case, link errors are predictable. Such predictability makes use of a short and fast feedback at the radio link layer rather than the end-to-end loop. The predictability was used in this work to determine the intervals in which the resynchronization side information should be sent in order to reduce the necessary rate.

As soon as the intervals with lower error probability can be predicted, this opens the possibilities for scheduling methods that send priority data in those intervals. However, additional delay is necessary for postponing the data; no rate increase is necessary in contrary to the retransmission mechanism. The simplest way of categorizing the data according to their importance is distinguishing by semantic information. I frames are more important than P frames, thus if this information is available at the scheduler, I frames may be postponed for transmission in time intervals with low error probability. The benefit is located in the range of approximately 1 dB in the static scenario. This is indeed a surprising gain since the static scenario is characterized by a rather small error probability. An improvement of approximately another 1 dB can be achieved if the importance of the data is determined finer. This can be achieved by the proposed distortion model parametrized by the amount of lost bits and picture area. Whereas in many research works scheduling of multiple users and/or services is investigated, in this work it has been shown, that a scheduling of a single stream of a single user still may provide means for robust transmission. Such scheduling can be regarded as a form of unequal error protection. Even if a layered architecture is beneficial for interactive and background services, that are not as critical to delay but extremely critical to bit errors, the performance of real-time video services can be essentially improved by relaxed layer architectures.

In this thesis, the focus was put on video streaming. Nevertheless, several proposed methods can also be used advantageously for video call and conferencing. Particularly beneficial for such conversational services are syntax analysis, VLC resynchronization combined with radio link packet CRC information, and the adaptive error concealment mechanism. All presented error detection and concealment strategies can be further deployed in video broadcasting and multicasting, except for the scheduling that utilizes link error prediction, since it requires a link quality feedback.

# Appendix A

# Abbreviations and Symbols

## A.1 List of Mathematical Symbols

| | |
|---|---|
| $a, \alpha$ | scalar variable or constant |
| $\underline{x}$ | vector |
| $\underline{\underline{X}}$ | matrix |
| $x(i, j)$ | element of matrix $\underline{\underline{x}}$ in the $i$th row and $j$th column |
| $f(x)$ | function, parameter is $x$ |
| $x[n]$ | sequence, parameter is $n$ |
| $\ell(x)$ | length of a sequence $x$ |
| $X$ | random variable |
| $\mathrm{E}\{X\}$ | expectation of random variable $X$ |
| $\underline{X}$ | random vector |
| $\mathcal{A}$ | set |
| $\lvert\mathcal{A}\rvert$ | cardinality of the set $\mathcal{A}$ |
| $\lvert\lvert\underline{c}\rvert\rvert_2$ | 2-norm of vector $\underline{c}$ |
| $\mathrm{P}\{\mathcal{A}\}$ | probability of event $\mathcal{A}$ |
| $p_X(x)$ | probability mass function of random variable $X$ for $X = x$ |
| $f_X(x)$ | probability density function of random variable $X$ for $X = x$ |
| $\mathbb{C}^{M \times N}$ | $(M \times N)$-dimensional space of complex numbers |
| $\mathbb{R}^{M \times N}$ | $(M \times N)$-dimensional space of real numbers |
| $\mathbb{N}^{+}$ | set of all positive integers |
| vec | operator of *vectorization* — a column wise stacking of a matrix into a vector |

## A.2 List of Abbreviations

| | |
|---|---|
| 3GPP | 3rd Generation Partnership Project |
| AAL | ATM Adaptation Layer |

| | |
|---|---|
| AC | Alternating Current |
| AM | Acknowledged Mode |
| ARIB | Association of Radio Industries and Businesses |
| ARQ | Automatic Repeat Request |
| AS | Application Server |
| ASO | Arbitrary Slice Ordering |
| ATIS | Alliance for Telecommunications Industry Solutions |
| ATM | Asynchronous Transfer Mode |
| AUC | Authentication Center |
| AVC | Advanced Video Coding |
| AWGN | Additive White Gaussian Noise |
| BEP | Bit Error Probability |
| BER | Bit Error Rate |
| BIAWGN | Binary-Input Additive White Gaussian Noise |
| BLER | Block Error Rate |
| BSC | Binary Symmetrical Channel |
| CABAC | Context Adaptive Binary Arithmetic Coding |
| CAVLC | Context Adaptive Variable Length Coding |
| CBR | Constant Bit Rate |
| CCCH | Common Control Channel |
| CCDF | Complementary Cumulative Distribution Function |
| CCSA | China Communications Standards Association |
| CCTrCh | Code Composite Transport Channel |
| CDF | Cumulative Distribution Function |
| CIF | Common Intermediate Format |
| CN | Core Network |
| CRC | Cyclic Redundancy Check |
| CS | Circuit Switched |
| CTCH | Common Traffic Channel |
| DC | Direct Current |
| DCCH | Dedicated Control Channel |
| DCH | Dedicated Channel |
| DCT | Discrete Cosine Transform |
| DL | Downlink |
| DP | Data Partitioning |
| DSCH | Downlink Shared Channel |
| DTCH | Dedicated Traffic Channel |
| ECDF | Empirical Cumulative Distribution Function |
| EIR | Equipment Identity Register |
| EPMF | Empirical Probability Mass Function |
| ETSI | European Telecommunications Standards Institute |

| | |
|---|---|
| FACH | Forward Access Channel |
| FB | FeedBack |
| FDD | Frequency Division Duplex |
| FEW | Force Even Watermarking |
| FIFO | First-In/First-Out |
| FIR | Finite Impulse Response |
| FLC | Fixed Length Code |
| FMO | Flexible Macroblock Ordering |
| FOD-US | First Order Derivative Uniform Smoothing |
| FOD-EA | First Order Derivative Edge Aware |
| GGSN | Gateway GPRS Support Node |
| GSM | Global System for Mobile Communications |
| HLR | Home Location Register |
| HSDPA | High Speed Downlink Packet Access |
| HSUPA | High Speed Uplink Packet Access |
| IDR | Instantaneous Decoding Refresh |
| IE | Information Element |
| IEC | International Electrotechnical Commission |
| IID | Independent, Identically Distributed |
| IP | Internet Protocol |
| ISO | International Organization for Standardization |
| ITU | International Telecommunication Union |
| JM | Joint Model |
| JSCC | Joint Source-Channel Coding |
| JVT | Joint Video Team |
| LAN | Local Area Network |
| LI | Length Indicator |
| LLR | Log-Likelihood Ratio |
| LogCh | Logical Channel |
| MAC | Medium Access Control |
| MB | MacroBlock |
| MDC | Multiple Description Coding |
| MPEG | Motion Picture Expert Group |
| MSC | Mobile Switching Center |
| MSE | Mean Square Error |
| MT | Mobile Termination |
| MTU | Maximum Transmission Unit |
| MV | Motion Vector |
| NAL | Network Abstraction Layer |
| NALU | Network Abstraction Layer Unit |
| NRI | NAL Reference Identification |

| | |
|---|---|
| NTSC | National Television Systems Committee |
| OSI | Open System Interconnection |
| PAL | Phase Alternation by Line |
| PC | Personal Computer |
| PCA | Principal Component Analysis |
| PDCP | Packet Data Control Protocol |
| PDF | Probability Density Function |
| PDU | Packet Data Unit |
| PHY | PHYsical Layer |
| PhyCh | Physical Channel |
| PI | Position Indicator |
| PMF | Probability Mass Function |
| POCS | Projection Onto Convex Sets |
| PPS | Picture Parameter Set |
| PS | Packet Switched |
| PSNR | Peak to Signal-to-Noise Ratio |
| QCIF | Quarter Common Intermediate Format |
| QoS | Quality of Service |
| QP | Quantization Parameter |
| QVGA | Quarter Video Graphics Array |
| RAB | Radio Access Bearer |
| RACH | Random Access Channel |
| RAN | Radio Access Network |
| RB | Radio Bearer |
| RBW | Relation Based Watermarking |
| RD | Rate-Distortion |
| RDO | Rate-Distortion Optimization |
| RGB | Red, Green, Blue |
| RLC | Radio Link Control |
| RNC | Radio Network Controller |
| RNS | Radio Network Subsystem |
| ROHC | RObust Header Compression |
| ROI | Region of Interest |
| RRC | Radio Resource Control |
| RS | Redundant Slices |
| RTCP | Real-Time Control Protocol |
| RTP | Real-Time Protocol |
| SAD | Sum of Absolute Differences |
| SAP | Service Access Point |
| SDU | Service data Unit |
| SECAM | Séquentiel couleur à mémoire |

| | |
|---|---|
| SEI | Supplemental Enhancement Information |
| SF | Spreading Factor |
| SGSN | Serving GPRS Support Node |
| SI | Spatial Information |
| SIF | Standard Interchange Format |
| SIM | Subscriber Identity Module |
| SIR | Signal to Interference Ratio |
| SM | Synchronization Mark |
| SMB | Submacroblock |
| SOD-US | Second Order Derivative Uniform Smoothing |
| SPS | Sequence Parameter Set |
| SVC | Scalable Video Coding |
| TB | Transport Block |
| TCP | Transmission Control Protocol |
| TDD | Time Division Duplex |
| TE | Terminal Equipment |
| TF | Transport Format |
| TI | Temporal Information |
| TM | Transparent Mode |
| TPC | Transmitter Power Control |
| TrCh | Transport Channel |
| TTA | Telecommunication Technology Association |
| TTC | Telecommunication Technology Committee |
| TTI | Transmission Timing Interval |
| UDP | User Datagram Protocol |
| UE | User Equipment |
| UEP | Unequal Error Protection |
| UICC | Universal Integrated Circuit Card |
| UL | Uplink |
| UM | Unacknowledged Mode |
| UMTS | Universal Mobile Telecommunications Network |
| USB | Universal Serial Bus |
| UTRAN | UMTS Terrestrial Radio Access Network |
| VBR | Variable Bit Rate |
| VCL | Video Coding Layer |
| VGA | Video Graphics Array |
| VLC | Variable Length Code |
| VLR | Visitor Location Register |
| WCDMA | Wideband Code Division Multiple Access |
| WM | WaterMarking |

# Appendix B

# Test Video Sequences

In order to evaluate and compare the performance of the different error resilience methods, representative video sequences are needed. There are several short reference video sequences, used by the standardization bodies as a reference. Throughout this work they are called *standard video sequences*. However, these sequences do not cover all types of content typical for video streaming services. Various types of content have different characteristics that may result in unequal performance of the techniques tested. Therefore, in this thesis additionally the video sequences are used that were recorded using a video camera or a PC TV card. The description and comparison of both the standard and *recorded video sequences* is presented in this appendix.

## B.1 Standard Video Sequences

The standard video sequences presented in this section are those, mostly used in scientific papers. They were obtained from various web pages (some of them not existing any more) in the YUV 4:2:0 format, without audio and were used for research purposes only.

### B.1.1 Foreman

The most popular among the reference video sequences is "foreman", the screenshots of which are shown in **Figure B.1**. This 400 frames long QCIF sequence contains a variety of the movement types and scenes. In the beginning, a static camera captures the head of a foreman with hardly any movement (probably waiting for an interview question). The background contains clear diagonal edges

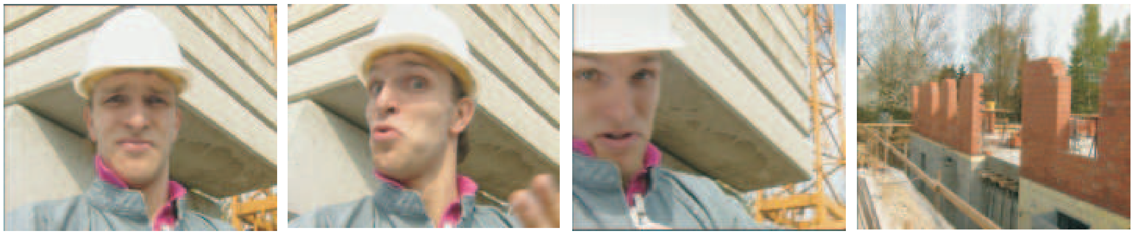


**Figure B.1**: Screenshots from "foreman" video sequence.

as well as some smooth areas. Later, the foreman starts a dynamic monologue, and makes fast head movements. His hand gesture is followed by a camera (fast panning, not exactly horizontal) and ends up in a scene containing a building site. Here, the camera stops but there is still some slight movement (due to missing stand).

### B.1.2 Carphone

The CIF video sequence "carphone" is 400 frames long. In **Figure B.2** some screenshots from this sequence are shown. An almost static camera captures a talking head in a car. The monologue is rather dynamic, the talking person moves his head fast. In the background, there is a back of seat covered by tissue with clear edges (stripes). Fast movement of the landscape can be seen in the car windows.

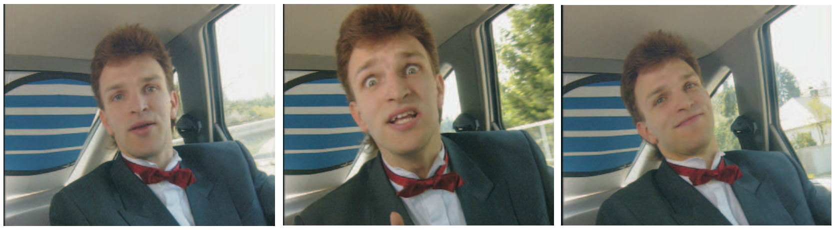


**Figure B.2**: Screenshots from "carphone" video sequence.

### B.1.3 Akiyo

The "akiyo" sequence is a 300 frames long QCIF video sequence, containing a moderator reading news as shown in the screenshots in **Figure B.3**. The camera is static, the background is rather smooth and does not change at all. The only movement in the sequence is the slight movement of the moderator's head — lips and eyes.

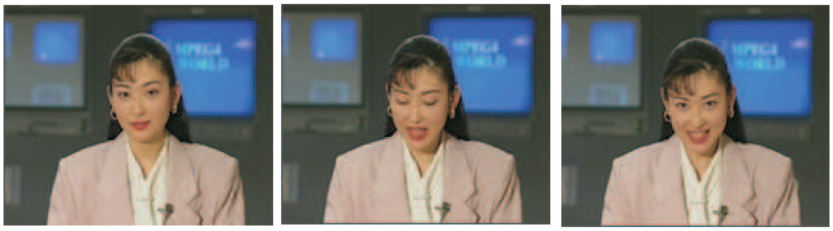


**Figure B.3**: Screenshots from "akiyo" video sequence.

### B.1.4 Mobile

The "mobile" sequence has QCIF resolution and the length of 300 frames. Some screenshots from this sequence can be seen in **Figure B.4**. An almost horizontally panning camera provides a view on a colourful wallpaper. In the foreground a model train is moving slowly tracked by the camera, it pushes a rolling ball. A vertically (up and down) moving calendar hangs on the wall.

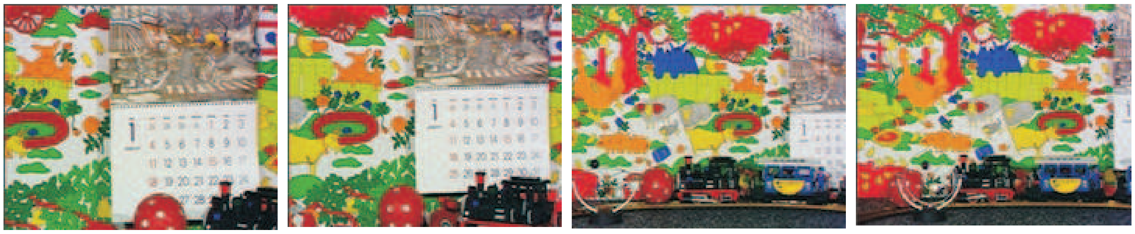


**Figure B.4**: Screenshots from “mobile” video sequence.

### B.1.5 Silent

The “silent” sequence has QCIF resolution and the length of 300 frames. Some screenshots from this sequence can be seen in **Figure B.5**. A static camera captures a lady communicating in deaf and dumb language, using rapid hand and head movements.

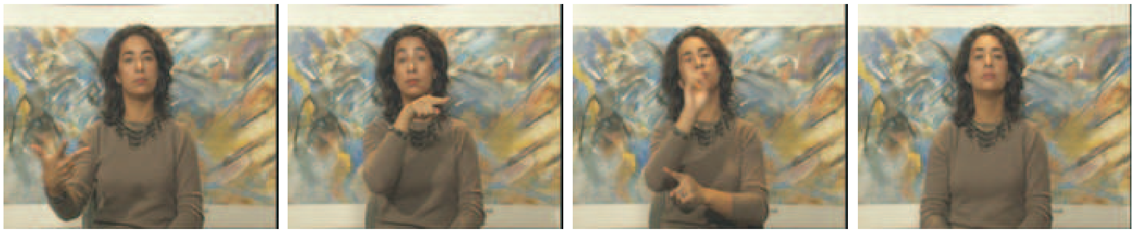


**Figure B.5**: Screenshots from “silent” video sequence.

### B.1.6 Glasgow

The “glasgow” sequence has QCIF resolution and the length of 100 frames. Some screenshots from this sequence can be seen in **Figure B.6**. Diverse shots taken by static or panning camera provide a guide over the city of Glasgow. The scenes are separated by 1 slow transition over 12 frames in the original 30 f/s frame rate and 3 cuts. The first scene is a view on a road in front of a building. Thereafter, several views of the People’s Palace winter garden are provided.

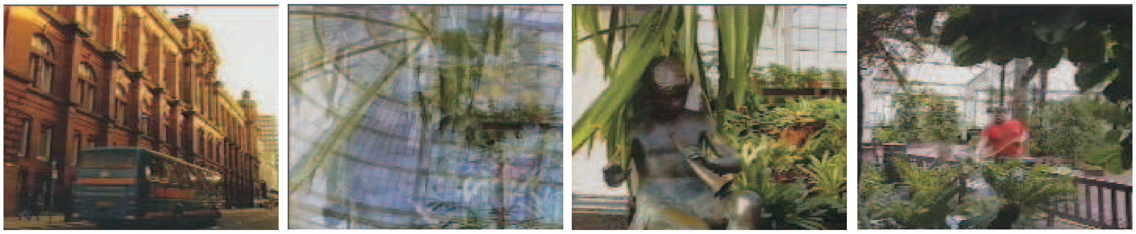


**Figure B.6**: Screenshots from “glasgow” video sequence.

## B.2 Recorded Video Sequences

THE recorded video sequences were recorded by means of a high-resolution digital camera, the captured sequences were captured using a PC TV card. The so obtained videos were converted in the raw 4:2:0 YUV format and downsampled to the desired spatial resolutions (QCIF in this specific case).

### B.2.1 Limbach

The QCIF video sequence "limbach" has a duration of 400 frames. It was recorded by a horizontally panning camera. The environment is a village with houses, garden fences and streets. There are no moving objects, the only movement is that of the camera. Some screenshots of this sequence are depicted in **Figure B.7**. The "limbach" sequence was created to represent the wide class of panorama sequences, occurring especially in weather cameras, tourist guide and surveillance applications.

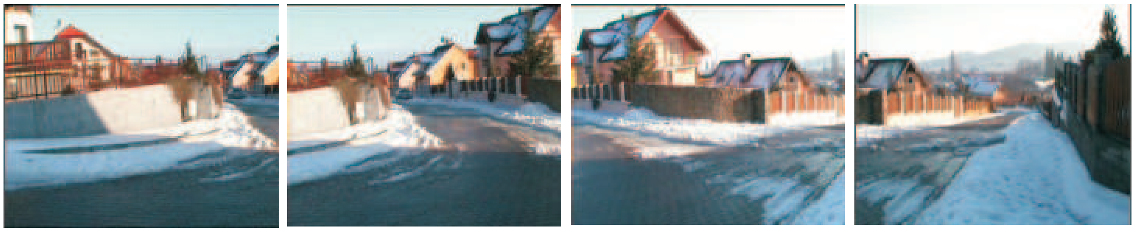


**Figure B.7**: Screenshots from "limbach" video sequence.

### B.2.2 Videoclip

The QCIF video sequence "videoclip" was captured from the TV. It is a video clip to a song. Within the 400 frames of the sequence, there are five scene cuts and four gradual changes — one short transition over two frames in the original 30 f/s frame rate and three longer transitions. The scenes contain static camera close-ups of the singer playing guitar, the vocal trio shot with a panning camera, the fast moving drummer, a musician playing synthesizer, and the singer dancing. There is a logo of the television on the right upper corner of each picture. This sequence represents a content type very popular for mobile video streaming and, at the same time, very demanding. Often, there are only several frames between two cuts in the video clips and movie teasers/trailers.

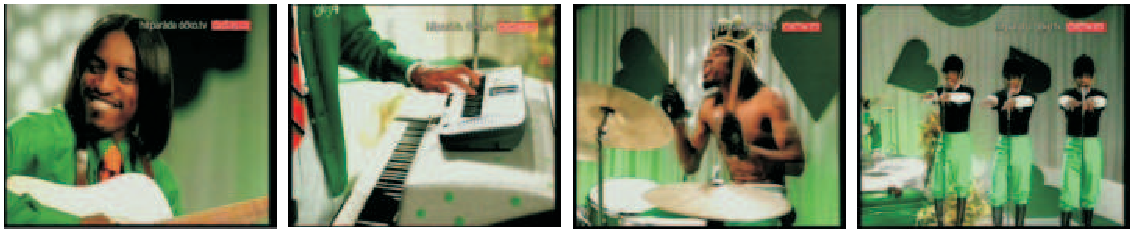


**Figure B.8**: Screenshots from "videoclip" video sequence.

### B.2.3 Squash

The video sequence “squash” was recorded by a static camera placed in one of the corners of a squash court as depicted in **Figure B.9**. The sequence is 400 frames long and has QCIF resolution. The players are changing their position during the game. The ball is very small and moves very fast, so that in some shots it is seen as a longer line.

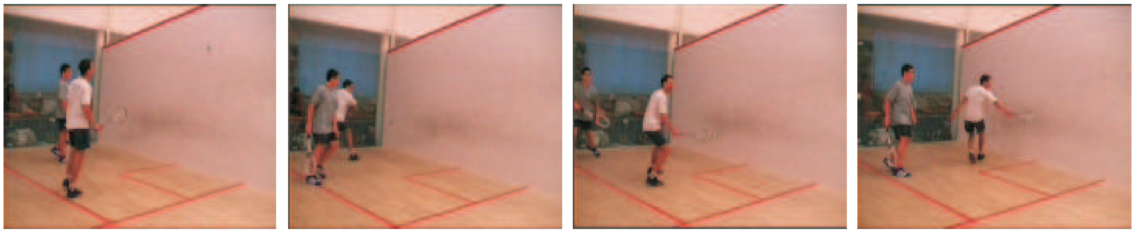


**Figure B.9**: Screenshots from “squash” video sequence.

### B.2.4 Soccer

The “soccer” video sequence utilized in this thesis is entirely shot by a wide angle horizontally panning camera following the ball. The 400 frame long sequence in the QCIF resolution contains the ball being not larger than 5–7 pixels. The prevailing color is green of the playground. The tribunes with audience cover only in average 7% of the picture area. The video captures a match ending-up by a goal. There are neither cuts nor scene changes. Some screenshots of this sequence can be seen in **Figure B.10**.

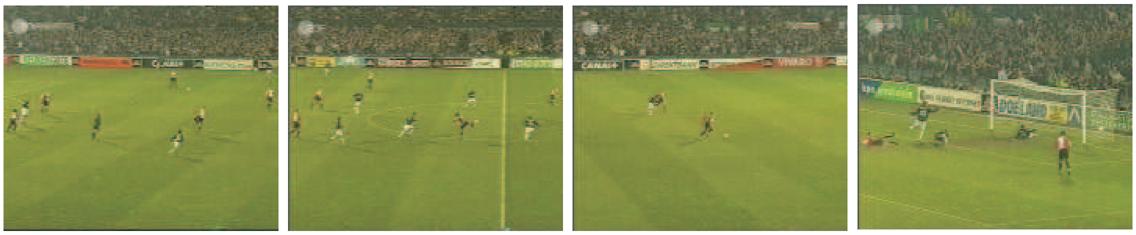


**Figure B.10**: Screenshots from “soccer” video sequence.

### B.2.5 Soccermatch

The “soccermatch” video sequence is a recorded soccer match. It was converted into both QVGA and QCIF resolutions, it contains 44000 frames. The “soccermatch” is a sequence consisting of various scenes separated by scene cuts and gradual changes (dissolves, zoom-in/outs and transitions). There are close-ups of the audience as well as of the players, wide angle camera shots under different angles, studio discussion shots etc. Some screenshots of this sequence can be seen in **Figure B.11**.

**Figure B.11**: Screenshots from “soccermatch” video sequence.

## B.3 Objective Measures of Sequence Characteristics

VIDEO sequences can be characterized by any movement features describing the speed and/or direction of the movement of their parts. Moreover, the content of particular pictures in the sequence can be described by the amount of edges corresponding to the gradient of the image. The standard for mean opinion score testing of streaming video [134] offers two metrics for spatial and temporal information presented in the following. These metrics are further used to characterize the above described video sequences.

### B.3.1 Spatial and Temporal Information

*Spatial Information* (SI) is a metric that determines the amount of the details in a video sequence. First, a Sobel mask (3.2) is applied to each pixel $(i, j)$ of the $n$th $N \times M$ large luminance frame $\underline{\underline{F}}_n$. According to (3.6), the magnitude $|\underline{\underline{G}}_n|$ consisting of elements $g_n(i, j)$ corresponding to the gradient field elements are calculated. The $\mathrm{SI}_n$ of the $n$th frame is then given by the estimated standard deviation of their magnitude:

$$\mathrm{SI}_n \triangleq \sqrt{\frac{1}{N \cdot M - 1} \sum_{i=1}^{i=N} \sum_{j=1}^{M} \left( g_n(i, j) - \overline{|\underline{\underline{G}}_n|} \right)^2}, \tag{B.1}$$

where $\overline{|\underline{\underline{G}}_n|} = \frac{1}{N \cdot M} \sum_{i=1}^{N} \sum_{j=1}^{M} g_n(i, j)$ is the mean of the magnitudes. The maximum value of $\mathrm{SI}_n$ within the whole sequence represents the spatial information

$$\mathrm{SI} = \max_n \mathrm{SI}_n. \tag{B.2}$$

*Temporal Information* (TI) characterizes the type of the movement in a video sequence. It is based upon the motion difference feature. For every time instance $n$, the luminance pixel values difference is calculated: $\underline{\underline{M}}_n = \underline{\underline{F}}_n - \underline{\underline{F}}_{n-1}$. The temporal information $\mathrm{TI}_n$ of the $n$th frame is then computed as the estimated standard deviation of its elements $m_n(i, j)$:

$$\mathrm{TI}_n \triangleq \sqrt{\frac{1}{N \cdot M - 1} \sum_{i=1}^{N} \sum_{j=1}^{M} \left( m_n(i, j) - \overline{\underline{\underline{M}}_n} \right)^2}, \tag{B.3}$$

with $\overline{\underline{\underline{M}}_n} = \frac{1}{N \cdot M} \sum_{i=1}^{N} \sum_{j=1}^{M} m_n(i, j)$ mean of the difference picture element values. The TI is given, again, as the maximum $\mathrm{TI}_n$ of the entire sequence:

$$\mathrm{TI} = \max_n \mathrm{TI}_n. \tag{B.4}$$

### B.3.2 Comparison of Video Sequences

The SI and TI calculated for the short video sequences introduced in this appendix are summarized in **Table B.1**. The values were calculated for the original sequence — uncompressed and with the original frame rate of 30 f/s. After compression and frame rate reduction, the values of both SI and TI are different[1]. Note that it does not make sense to characterize the "soccermatch" sequence in terms of SI and TI, since it consists of a multitude of totally different scenes. Hence, it is not present in the list above. Note also that both SI and TI correspond to the *maximum* of spatial and temporal information, respectively. Only the maximum does not provide much information about the distribution of the spatial and temporal information within the entire sequence. Other statistics as for example mean and/or median could provide information more relevant to the character of the whole sequence. In this theses, however, the standardized statistics were utilized.

| Sequence | SI | TI |
|---|---|---|
| foreman | 122.60 | 42.53 |
| carphone | 104.24 | 33.35 |
| akiyo | 91.84 | 5.99 |
| mobile | 183.90 | 22.03 |
| silent | 98.73 | 14.28 |
| glasgow | 128.15 | 93.76 |
| limbach | 154.16 | 24.86 |
| squash | 96.25 | 12.22 |
| videoclip | 180.44 | 91.89 |
| soccer | 99.86 | 26.66 |

**Table B.1**: Values of SI and TI of the presented short video sequences, calculated for the uncompressed original.

More details are presented in **Figure B.12**, where the scatter plot of $\mathrm{SI}_n$ and $\mathrm{TI}_n$ values is presented for all chosen short video sequences. It can be seen that the chosen video sequences cover a wide range of the spatial and temporal information. The recorded sequences represent new types of the content. Especially the "videoclip" due to its scene changes and cuts contains a large scale of $\mathrm{SI}_n$ and $\mathrm{TI}_n$ values. The scene cuts and changes are clearly visible (the higher $\mathrm{TI}_n$ values). The highest $\mathrm{TI}_n$ value achieved the "glasgow" sequence due to its cuts separating completely different scenes. The video sequences with panning camera usually cover a narrow range of the $\mathrm{TI}_n$ values as can be seen for the "limbach" and partly the "soccer" sequences. However, the $\mathrm{SI}_n$ values of "soccer" are smaller than "limbach" mainly due to the spatial smoothness of the green playfield. The "akiyo" sequence provides the smallest variance of both the $\mathrm{SI}_n$ and the $\mathrm{TI}_n$ values — it is thus e.g. not as critical for error concealment. Not surprisingly, the highest $\mathrm{SI}_n$ values belong to the "mobile" sequence, mainly due to the wallpaper background containing a lot of details. The value range of $\mathrm{SI}_n$ and $\mathrm{TI}_n$ for the "soccermatch" sequence can be seen in **Figure B.13**. This long video sequence covers the entire value

---

[1] The difference between the SI and TI or similarly obtained parameters can thus also be utilized to measure the quality degradation.

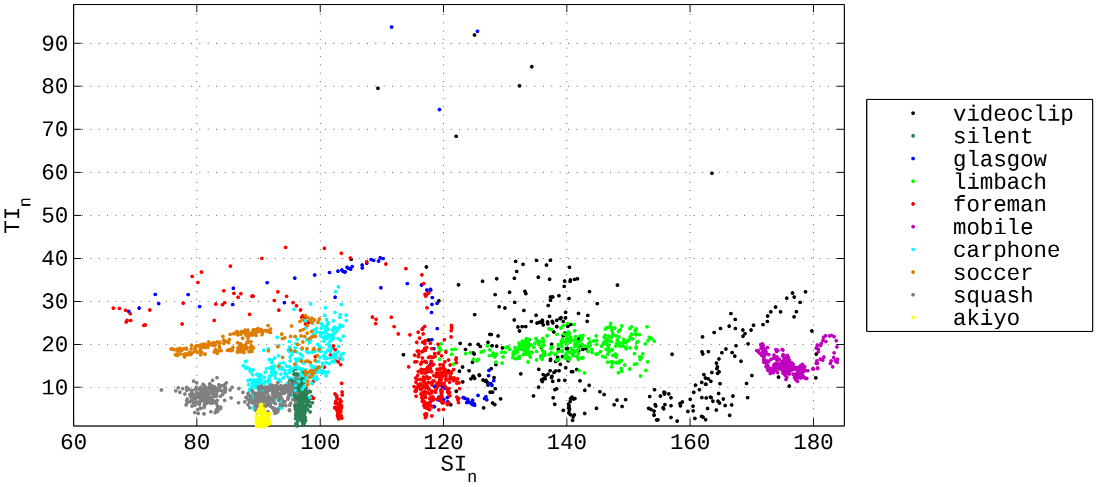


**Figure B.12**: The values of $\mathrm{SI}_n$ and $\mathrm{TI}_n$ for all the presented short video sequences, calculated for the uncompressed original.

range of $\mathrm{SI}_n$ and $\mathrm{TI}_n$ belonging to all presented short video sequences. Hence, it is perfectly suitable to simulate the wide variety of contents if longer simulations are necessary for better statistics. Again, the scene cuts and fast scene changes can clearly be seen in the upper part of the $\mathrm{TI}_n$ values.

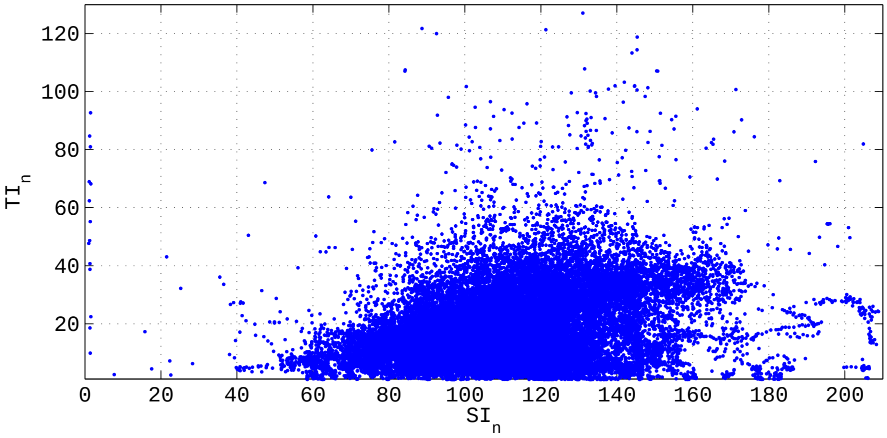


**Figure B.13**: The values of $\mathrm{SI}_n$ and $\mathrm{TI}_n$ for the presented “soccermatch” video sequence, calculated for the uncompressed original.

# Appendix C

# UMTS Overview

THE Universal Mobile Telecommunications System (UMTS) is the first and so far the only 3rd generation wireless mobile systems in Europe, working in the licensed 2 GHz band. In UMTS standard, three possible modi can be found: Frequency Division Duplex (FDD), Time Division Duplex (TDD) and TDD with low chip rate. Nowadays, predominantly the UMTS FDD mode is deployed by many mobile operators all over the world. Therefore, in this brief description only FDD will be considered.

The UMTS brought data rates up to 2 Mbit/s, which allowed to provide new services, for instance video conferencing, audio and/or video streaming, web browsing, gaming and more. The UMTS standard exists in various releases, each of them including new features. The terminals today (in the time of writing the thesis) implement release 4 and 5 features. The most important additional feature in release 5 is the High Speed Downlink Packet Access (HSDPA), supporting data rates up to 14 Mbit/s in the downlink direction [1]. Release 6 contains many improvements, among them also the High Speed Uplink Packet Access (HSUPA), aiming to achieve higher throughput in uplink direction and providing so a complement to HSDPA. Other improvements and architecture changes are studied for future releases.

## C.1 UMTS Architecture

THE UMTS network consists of a number of network elements having predefined functionality. They are grouped into the following three high-level system architecture units [135]:

- **Core Network** (CN) responsible for switching and routing calls and data connections to external networks,
- **UMTS Terrestrial Radio Access Network** (UTRAN) providing radio related functionality,
- **User Equipment** (UE) responsible for interfacing with user and with radio.

Whereas both the UE and UTRAN consist of new specific protocols, adapted to the WCDMA physical layer, the CN is adopted from the 2nd generation Global System for Mobile Communications (GSM). The UE with UTRAN and UTRAN with CN are connected via standardized interfaces Uu and Iu,

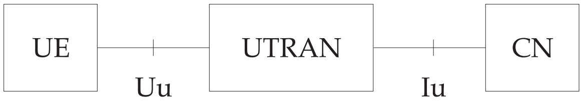


**Figure C.1**: UMTS architecture.

respectively, as illustrated in **Figure C.1**. UTRAN controls the transmission over the wireless interface by means of Radio Network Controller (RNC) network elements, each of which again controls a multitude of base stations (in UMTS denoted NodeB) connected by the standardized interface Iub. The area covered by a NodeB is called *cell*. A cluster consisting of an RNC and its controlled NodeBs is called Radio Network Subsystem (RNS). This architecture is illustrated in **Figure C.2**. In UMTS, a UE can have simultaneous connection to more than one NodeBs, the NodeBs may or may not be controlled by the same RNC. This is called *soft handover*. Each NodeB can control several antennas serving several sectors. If then the UE is connected to more than one of the sectors, it is said to be in *softer handover*. Since the NodeBs are not interconnected, the RNC is the only entity having information about the inter-cell interference. Thus, the RNC performs the most important radio resource management tasks including scheduling, call admission control, congestion and flow control etc. RNCs are interconnected via a standardized interface called Iur.

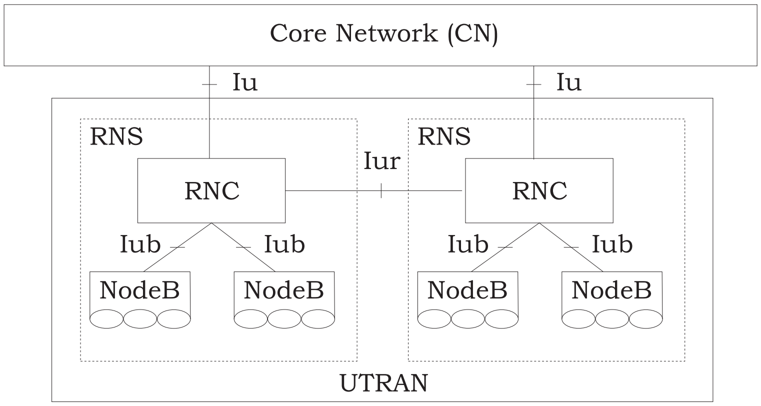


**Figure C.2**: UTRAN architecture.

The UE consists of three main parts — the Terminal Equipment (TE), the Mobile Termination (MT) and the Universal Integrated Circuit Card (UICC). The TE provides end-user application functions and terminates the upper layers. It may be coupled to a non-UMTS entity (e.g. mobile connected to a PC via BlueTooth). The MT terminates the radio transmission. The UICC is the user subscription dependent part of the UE, it contains the Subscriber Identity Module(s) (SIM).

The CN is divided in circuit switched (CS) and packet switched (PS) domains. Some of the circuit switched elements are the Mobile Switching Centre (MSC), the Visitor Location Register (VLR) and the Gateway MSC. Packet switched elements are the Serving GPRS Support Node (SGSN) and the Gateway GPRS Support Node (GGSN). Some network elements, like the Equipment Identity Register (EIR), the Home Location Register (HLR), the VLR and the Authentication Center (AUC) are shared

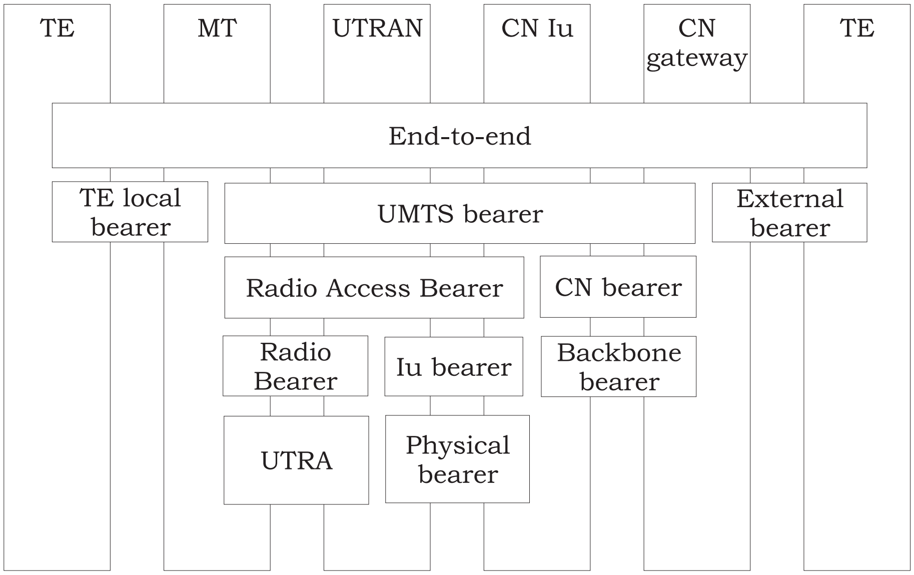


**Figure C.3**: QoS levels in UMTS.

by both domains. The architecture of the Core Network may change when new services and features are introduced, resulting in new network elements to be added to the CN. More information on CN can be found in [1]. The Asynchronous Transfer Mode (ATM) is defined for the UMTS core transmission. The ATM Adaptation Layer type 2 (AAL2) handles a CS connection and the packet connection protocol AAL5 is designed for data delivery.

## C.2 Quality of Service in UMTS

In **Figure C.3** an overview of different levels of QoS in UMTS is shown. Network services are considered end-to-end, i.e., from a TE to another TE. An end-to-end service may have a certain QoS that is provided for the user of a network service. It is the user who decides whether (s)he is satisfied with the provided QoS or not. To realize a certain network QoS, a Bearer Service with clearly defined characteristics and functionality is to be set up from the source to the destination of a service. A bearer service includes all aspects to enable the provision of a contracted QoS. These aspects are among others the control signalling, user plane transport and QoS management functionality. The Radio Access Bearer (RAB) defines the service, which UTRAN provides for the CN. The following four service classes are defined in UMTS: conversational, streaming, interactive, and background class. Their characteristics are listed in **Table C.1**.

The attributes of these classes and their limits can be found in [136]. Note that conversational and streaming services belong to guaranteed services, where guaranteed bit rate and transfer delay can be defined as a QoS attribute.

| Class | Characteristic | Applications |
|---|---|---|
| Conversational | • preserve time relation (variation) between information entities of the stream.<br>• conversational pattern (stringent and low delay) | voice/video call, conferencing |
| Streaming | • preserve time relation (variation) between information entities of the stream. | voice/video streaming |
| Interactive | • Request response pattern<br>• Preserve payload content | web browsing |
| Background | • Destination is not expecting the data within a certain time<br>• Preserve payload content | e-mail,<br>background file download |

**Table C.1**: UMTS QoS classes.

## C.3 Protocol Architecture

THE end-to-end protocol architecture depends on the transported service. Before being transmitted on the radio interface, video packets are typically encapsulated in RTP/UDP/IP protocols at the Application Server (AS) and sent over various external and CN interfaces.

### C.3.1 Radio Interface Protocols

The 3GPP standard distinguishes between the user plane protocols, carrying the user data and the control plane protocols, carrying the signalling information. In **Figure C.4** the protocol architecture of the UTRAN radio interface is shown. Note that this architecture includes three network elements (UE, NodeB, RNC) and correspondingly two/three interfaces (wireless Uu, wired Iub and possibly Iur). User plane and control plane protocols of the Iub and Iur interface are omitted here. The PHY denotes the physical layer of the radio interface; its functions are related to the nature of the wireless mobile channel in UMTS. The PHY is based on the wide-band CDMA (WCDMA) technology.

Above the PHY layer, the Medium Access Control (MAC) layer can be found. The main release-99/4 MAC [137] function is the mapping of logical channels (LogCh) onto the transport channels (TrChs), including multiplexing. This requires also selection of the appropriate Transport Format (TF) for each TrCh depending on the instantaneous source rates of the logical channels. The TF is selected with respect to the TF combination set defined by admission control for each connection at the radio access bearer setup. The functions of MAC further include priority handling for different data flows of one UE, dynamic scheduling for shared and common transport channels, traffic volume monitoring (used to trigger RAB reconfiguration if necessary), transport channel type switching and ciphering (for transparent mode RLC).

The Radio Link Control (RLC) protocol [138] provides segmentation and retransmission services

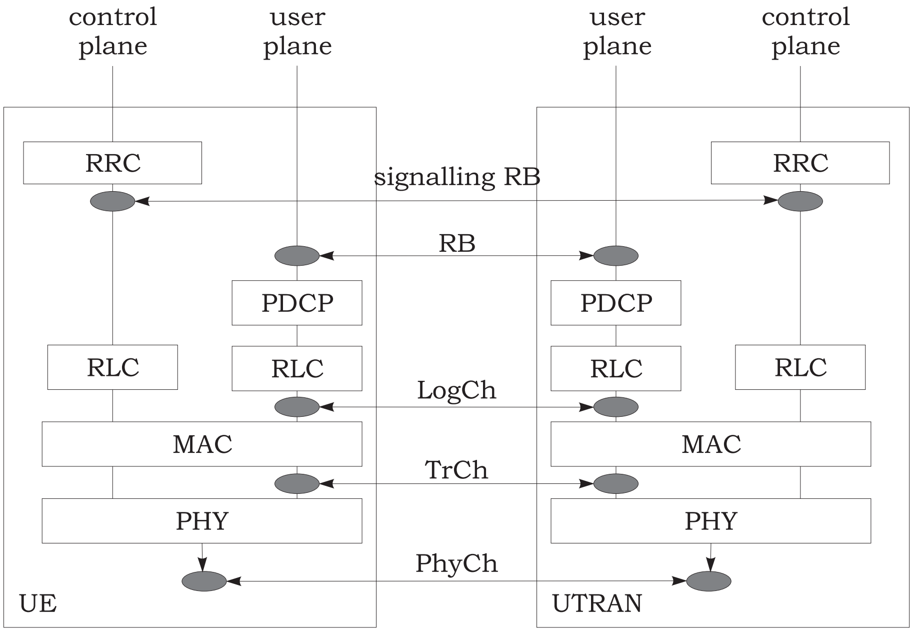


**Figure C.4**: UTRAN protocol model. The dark ellipses denote Service Access Points (SAP).

for both, control and user plane. The RLC, as well as MAC and PHY, is configured by the Radio Resource Control (RRC) protocol. The RLC can operate in three modes: transparent mode (TM), acknowledged mode (AM) and unacknowledged mode (UM). The TM is used for CS data; the RLC in TM does not add any overhead, it does not provide any retransmissions, usually no segmentation applies; if it is applied, it has to be negotiated at radio bearer setup. In UM no retransmissions are performed. At the transmitter side, a timer based discard is performed for packets that were not transmitted within the specified time. To facilitate this, the RLC adds an eight bit long header containing the sequence number. The AM provides an automatic repeat request (ARQ) mechanism to correct the transmission errors. In summary, the RLC fulfills the following functions: segmentation and reassembly (typically AM/UM RLC packets are 320 or 640 bit long), concatenation and padding, in-sequence delivery (optional, requires reordering buffer), flow control and ciphering.

The Packet Data Convergence Protocol (PDCP) [139] exists in the user plane only, and only for the PS CN domain. Its main task is to perform the compression of the RTP/UDP/IP header, to save radio resources. The RTP/UDP/IP header can be 40 (IPv4) or 60 (IPv6) byte long, which is e.g. for small speech packets longer than the payload.

The RRC protocol [140] is a control plane protocol that controls the UTRAN configuration. It is responsible for the RAB setup and reconfiguration, physical layer measurements, configuration of system information etc.

### C.3.2 Channels

The concept of a fixed set of configurable logical (LogCh), transport (TrCh) and physical (PhyCh) channels in UMTS provides means for describing the structure of data transported at different protocol layers. The channels can facilitate communication in the direction from UTRAN to UE, called downlink (DL) and/or from UE to UTRAN, called uplink (UL).

Logical channels determine what kind of information (control or user traffic) is transmitted. Logical channels in UMTS are e.g. the Dedicated Traffic Channel (DTCH), Dedicated Control Channel (DCCH), Common Traffic Channel (CTCH) and Common Control Channel (CCCH).

Transport channels determine the type of the data transport: dedicated (designated to one user only), shared (each user has access to the channel at certain time), or common (used by all user). The following transport channels are defined in UMTS Rel-99/4 for transport of user data: Dedicated Channel (DCH), Downlink Shared Channel (DSCH), Forward Access Channel (FACH), Random Access Channel (RACH). The most relevant channel for unicast transport of multimedia is the DCH, existing in both UL and DL direction, carrying the data dedicated to one user only. The radio resource management for the DCH is entirely performed in the RNC; the NodeB implements only the physical layer processing for DCH. The DSCH was specified in order to support asymmetric traffic, since there is higher data volume in DL than in UL. This is the typical case for many packet-oriented applications like web browsing, file download, or streaming (audio or video). The DSCH exists in DL only. At present it is not used in practice. In Release 5, an alternative high-speed DL shared channel is specified within the HSDPA extension. The primary function of the RACH is the initial access. The initial access is performed by sending the channel allocation request in a random time. The power is controlled by ramping-up until an answer is obtained or the power limit is exceeded. Thus, the RACH only exists in uplink and is not collision free. If a collision occurs, a back-off mechanism reduces further collisions. The FACH is a common channel used to carry the answer to initial access and resource allocation as well as system broadcast in the connected mode. It exists in DL only. Both, RACH and FACH can also be used to transfer small amounts of data (in [141] and [142] up to 32 kbit/s data rates can be found).

Physical channels refer to a particular configuration of the physical layer used for the transport of data.

### C.3.3 Packetization

Each IP packet, that enters the UTRAN RLC layer located in the RNC, is segmented and mapped onto the transport channel by the MAC protocol layer and on the physical layer as shown in **Figure C.5**. Before the segmentation of an IP packet, the PDCP may perform robust header compression (ROHC)

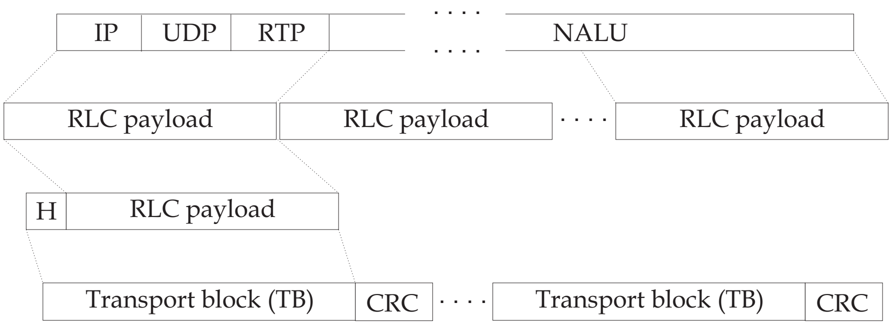


**Figure C.5**: Packetization example of a video slice for transmission over the UMTS radio interface.

to reduce the overhead of the end-to-end protocol header (RTP/UDP/IP) at the radio interface. In case of a bearer with a data rate below 384 kbit/s, usually a 320 bit (40 byte) payload within the RLC packets is used; for higher data rates a 640 bit long payload may be used. For PS bearers, the RLC of UTRAN can work in acknowledged mode allowing the RLC packet retransmissions or in unacknowledged mode allowing only for error detection but not for feedback. The header size for the RLC AM packet is 16 bit, whereas the RLC UM header is 8 bit long. By adding these RLC headers to the RLC packets each RLC packet becomes a transport block. To each TB a Cyclic Redundancy Check (CRC) is attached where the size of the CRC is configurable and may be 0,8,12,16 or 24 bit long [143]. The number of transport blocks in a TB set, interleaved over one Transmission Timing Interval (TTI) is defined by the TF. Transmission Timing Interval determines the periodicity at which the TB set is delivered from MAC to physical layer. It is always a multiple of the minimum interleaving period (i.e., 10 ms corresponding to one radio frame). After the segmentation/concatenation into the code blocks, the bit stream is encoded by a channel code. For packet oriented applications usually a turbo code is used with a code rate of 1/3, which can further be punctured to match the rate with the physical resources. After the mapping and radio frame segmentation, the bit sequence is further spread, scrambled, modulated, shaped and transmitted.

# Appendix D

# Channel Error Models

PROPER modeling of the transmission channel is essential to perform meaningful simulations. According to the simulation purpose, the channel models represent the statistical error behavior of one or more protocol layers.

The bit (physical) layer, link layer, and transport layer error models used throughout this thesis are introduced in this appendix. Bit error models are used to describe the errors in the bitstream after the channel decoding. The bit error characteristic is known at the receiver only if the original content of the received data sequence is known. In practice, errors in the Rel-99/4 UMTS are detected first at the RLC layer by a CRC applied to RLC packets with configurable length. A transport layer error model describes end-to-end packetloss statistics. It relays on the checksum attached to the transport layer (e.g. UDP or TCP) packets.

## D.1 Bit Error Models

THE two bit error models in this thesis (see Chapter 2) are both memoryless, since each channel output only depends on its instantaneous input, but not on the previous or future inputs. This is an assumption that is justified by using interleaving at the physical layer. The first error model is the Binary Symmetrical Channel (BSC) describing a binary memoryless channel with non-zero probability of a bit inversion. The second model is the Binary Input Additive White Gaussian Noise (BIAWGN) useful if a soft-input method can be applied on the received values.

### D.1.1 Binary Symmetrical Channel (BSC)

In coding theory, a binary symmetric channel is an idealized model of a communication channel over which bits (zeros or ones) are sent. In a BSC, the probability $p$ of a one detected as zero and of a zero detected as one is assumed to be the same — hence the channel is symmetric with respect to errors. Consequently, the probability of a bit sent over a BSC being correctly received is $(1-p)$, and this probability is independent of which bit has been sent. Assuming that $p$ is known to the receiver, we may without loss of generality assume that $p \leq 1/2$ since otherwise we may simply invert the received symbols resulting in error probability of $1-p < 1/2$.

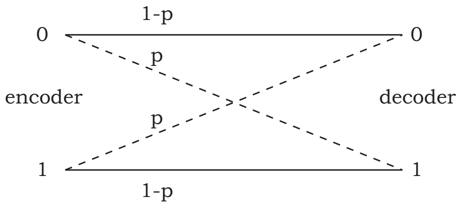


**Figure D.1**: Binary symmetric channel.

### D.1.2 Binary Input Additive White Gaussian Noise (BIAWGN)

BIAWGN is a binary-input continuous-output channel based on the model shown in **Figure D.2**. The random variable $X[n] \in \mathcal{A}$ represents an input symbol at the time $n$ taking the value $s_0$ or $s_1$ from a binary alphabet $\mathcal{A} = \{s_0, s_1\}$. To each input symbol $X[n]$, a Gaussian distributed noise $V[n] \sim \mathcal{N}(0, \sigma^2)$ is added:

$$f_V(v) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{v^2}{2\sigma^2}}. \tag{D.1}$$

The output is given by $Y[n] = X[n] + V[n]$, with $Y[n] \in \mathbb{R}$. Such a channel is characterized by the conditional probability density function relating the real-valued output to both possible inputs. The noise is white, i.e. $V[i]$ and $V[j]$ are uncorrelated for each $i \neq j$, i.e. the channel is memoryless. After applying a hard decision on a BIAWGN channel, the BSC channel model is obtained.

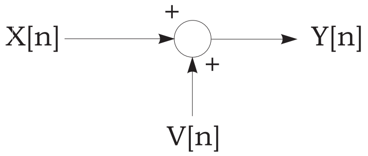


**Figure D.2**: Model for additive white Gaussian noise.

## D.2 Link Layer Error Models

THROUGHOUT this thesis (cf. Chapter 4), link layer error models designed for the UMTS Dedicated CHannel (DCH) in downlink are used, corresponding to the typical video streaming configuration. As shown by means of the measurements performed and presented in this section, the errors at UMTS DCH have a bursty nature. Hence, memoryless models are absolutely unsuitable (as shown in [79]) to model the errors in UMTS DCH channel. The Gilbert and Gilbert-Elliott error models, introduced in this section, are often used for modeling of bursty errors/losses. However, they reflect the distribution of neither error bursts nor error free intervals (gaps) in UMTS DCH. Therefore, two so called models, based on measured traces, are presented and their superior performance is demonstrated. These models are called "Karner" models throughout this thesis.

### D.2.1 Gilbert Model

The widest known burst error model is the Gilbert model [144]. It is a two-state model that performs transitions between a *good state* and a *bad state* according to the state transition probabilities $p_{01}$ and $p_{10}$ as shown in **Figure D.3**. In the good state, the error/loss probability $p_g$ is zero. In the bad state, an error/loss occurs with an independent error/loss probability $p_b$. Thus, to fully describe the Gilbert model, three parameter are needed: $p_{01}, p_{10}$, and $p_b$. It is often (incorrectly) assumed [145] that the Gilbert model bad state corresponds to an error/loss state, i.e. that $p_b = 1$. Such model would correspond to a simple *two-state Markov model*, allowing for consecutive errors/losses only. The clusters containing errors/losses separated by very short or no gaps cannot be modeled. A two-state Markov model is fully described by its two transition probabilities $p_{01}$ and $p_{10}$.

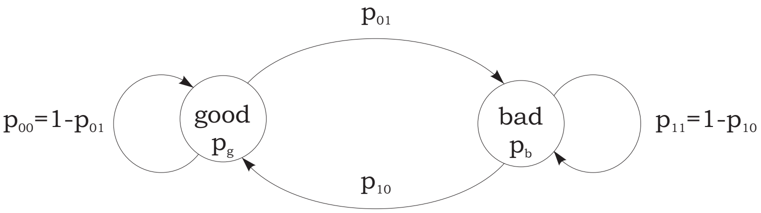


**Figure D.3**: General scheme for Gilbert-Elliot, Gilbert ($p_g = 0$), and the two-state Markov model ($p_g = 0, p_b = 1$).

The Gilbert model was later extended by Elliot [146], resulting in what is called today the *Gilbert-Elliot model*. The Gilbert-Elliot model assumes a low, but non-zero independent probability $p_g > 0$ of error/loss even in the good state. Thus, it needs four parameters to be fully described.

### D.2.2 Improved Models

The Gilbert-Elliot, Gilbert, and the two-layer Markov model are not capable of modeling every error distribution. The most realistic and precise way of modeling the link-layer error statistics in a UMTS is using the traces from a live network. However, the length of the traces is usually limited and requires considerable storage capacity. Especially, for long simulations a model is desired that generates the error trace in the run time. This requirement is met by the Karner models, that are based on the traces of the live UMTS network.

#### D.2.2.1 Measured Traces

The traces employed in this thesis were obtained from measurements in three live UMTS networks operated by three different companies in the city center of Vienna, Austria. To obtain the traces, a packet switched (PS) radio access bearer (RAB) of 384 kbit/s was configured in the beginning of the measurements. The RLC was working in acknowledged mode. As a transport channel in downlink a DCH has been used. A transport block (TB) size of 336 bits has been selected, the spreading factor (SF) was 8, TTI was 10 ms with 12 TBs per TTI and a turbo code was used. The BLER (BLock Error Rate) quality target value for the outer loop transmit power control was set to 1% in all three networks. The network was configured to perform RAB switching (to a 128 kbit/s and 64 kbit/s RAB) triggered by the persisting decrease of the channel quality (for details see [147]). The 128 kbit/s bearer used a SF of 16 and a TTI of 20 ms with eight transport blocks per TTI. The 64 kbit/s bearer was adjusted to a SF of 32, the TTI was 20 ms and there were four transport blocks transmitted within each TTI.

For the measurements, a UDP data stream with a bit rate of 360 kbit/s (372 kbit/s including UDP/IP overhead) in downlink was used. The data was sent from a PC located at the Institute of Communications and Radio Frequency Engineering at the Vienna University of Technology to a notebook using a UMTS terminal as a modem. As depicted in **Figure D.4** the UDP data stream originates from the PC running over the university Local Area Network (LAN) — ethernet, internet, UMTS core network and over the UMTS radio interface to a UMTS mobile phone, which is connected via USB (Universal Serial Bus) to the notebook. A WCDMA TEMS Mobile Motorola A835[1] from Ericsson [148] was used as terminal. On the notebook, the measurements have been captured by "TEMS Investigation WCDMA 2.4" software produced by Ericsson as well. This software is capable of presenting the CRC information of the received transport blocks. By parsing the export file of "TEMS Investigation WCDMA 2.4", the CRC information can be extracted, saved and used as an RLC error trace.

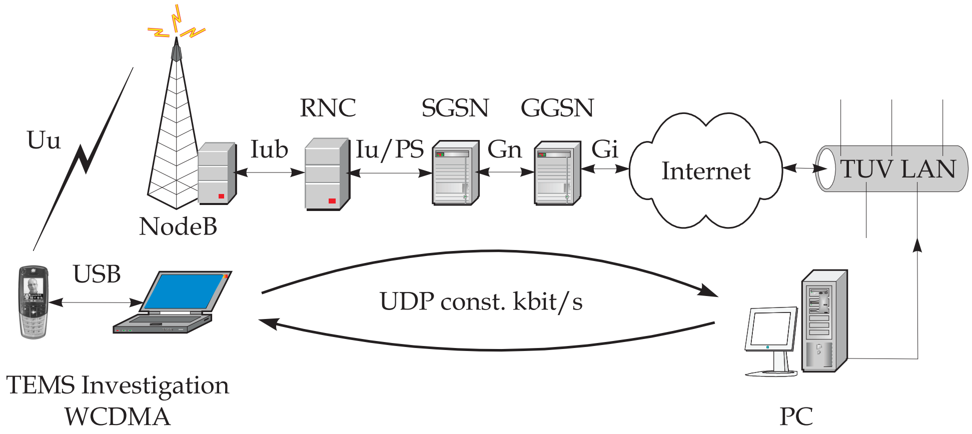


**Figure D.4**: Setup for measurement of link layer error traces at UMTS DCH.

The measurements were performed at various conditions, covering the most important scenarios in which a user could utilize the UMTS packet switched services. After investigating the resulting traces and their characteristics, similarities between the statistics of traces obtained under different conditions have been found [149]. This led to defining the two groups – *static scenario* and *dynamic scenario*.

The measurements for the static scenario were performed in a room of the Institute of Communications and Radio Frequency Engineering at the Vienna University of Technology. For these measurements the UMTS terminal was lying on the table in a typical office environment. Due to few movement of persons or other objects (scatterers) around the mobile station, there were only little

---

[1] A modified Motorola A835 for TEMS data logging. Modifications were only performed in order to provide internal measurements via the interface.

variations in the channel. The dynamic scenario includes “small scale movement”, “walking indoor”, “tram 2”, “car Vienna/Guertel” and “car highway Vienna-Schwechat”. The “small scale movement” measurements were performed by a person sitting at the table and randomly tilting and moving the UMTS mobile with his hands. In the “walking indoor”, the measurements were obtained while walking around in the building of the Institute of Communications and Radio Frequency Engineering. Static scenario together with “small scale movement” and “walking indoor” are indoor scenarios whereas the following three cases were measured outdoor — in the tramway number two going round the city center of Vienna (“tram 2”) and going by car either on the street in Vienna called Guertel (“car Vienna/Guertel”) with moderate speed or on the highway from Vienna to Schwechat (“car highway Vienna-Schwechat”) with higher speeds.

#### D.2.2.2 UMTS RLC Karner Models

The goal in modeling the error characteristics of the UMTS DCH is to generate a sequence of transport blocks with the statistics of gap lengths and burstlengths corresponding to the values measured in given scenarios (as described in the previous section). The Karner models presented in [149] and [150] fulfill this requirement. The Karner models fit well the UMTS DCH characteristics in both static and dynamic scenario unlike other typically used error models (e.g. Gilbert, Gilbert-Elliot (cf. Section D.2.1), or Fritchman model [151]).

**One-Layer Karner Model**

The one-layer Karner model is described by a renewal process with two states (see the schematic illustration in **Figure D.5**), in the first one a gap is generated, in the second one a burst is generated. There is no dependence between subsequent gaps, subsequent bursts or neither between a gap and the following burst.

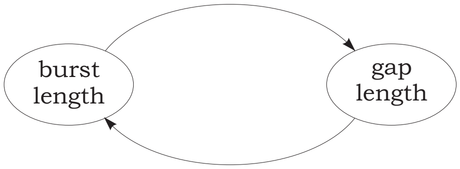


**Figure D.5**: Schematic representation of the one-layer Karner model.

The gap length is generated by a Weibull distribution with $a$ and $b$ being the scale and shape parameters, respectively:

$$f_X(x) = ba^{-b}x^{b-1}e^{-(\frac{x}{a})^b}. \tag{D.2}$$

To generate the gap length reflecting the measured distribution, two sets of parameters: $a^{gap}_{short}$ and $b^{gap}_{short}$ for ‘short’ gaps (smaller than or equal to 12 TBs), and $a^{gap}_{long}$ and $b^{gap}_{long}$ for the ‘long’ gaps (much greater than 12 TBs). These are the four parameters necessary to describe the gap length generating part of the one-layer Karner model. Additionally, a fifth parameter is needed — the probability of a short gap $P^{gap}_{short}$. The probability of a long gap can then be correspondingly calculated as $1 - P^{gap}_{short}$. After each calculation of a gap length, the state is changed (with probability equal to one), and a burstlength has to be generated. The burstlengths can be either Weibull distributed (with parameters $a^{burst}$ and $b^{burst}$) or deterministic. The deterministic values for the 'short' and the 'long' bursts follow from the measured distribution where distinctive steps at these values (12 and 24 TBs for the 384kbit/s RAB) can be observed (see **Figure D.8**). These steps are caused by the interleaving period. Thus, for modeling of burstlengths in the one-layer Karner model, four parameters are needed: $a^{burst}$ and $b^{burst}$ for the Weibull distribution and the probability of the two states with the 'short' and the 'long' deterministic burstlength $P^{burst}_{short}$ and $P^{burst}_{long}$. The probability of the Weibull distributed burstlength is then given by $1 - P^{burst}_{short} - P^{burst}_{long}$. After a burstlength was generated, the gap length has to be generated again — the model state is changed again with the probability of one.

### Two-Layer Karner Model

In practice, statistical dependency between subsequent gaps, bursts and also between a gap and the following burst was observed [150]. The two-layer Karner model models these dependencies by means of the upper layer two-state Markov chain as illustrated in **Figure D.6**.

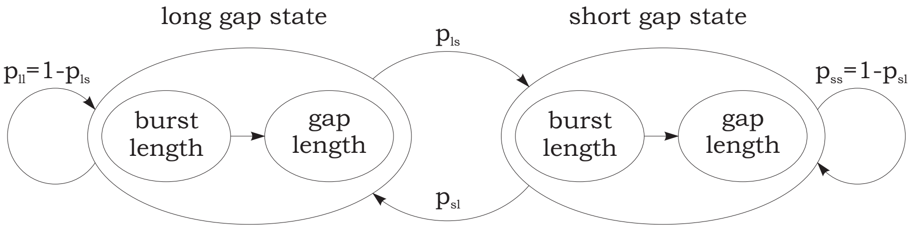


**Figure D.6**: Schematic representation of the two-layer Karner model.

The model has one 'short' gap state (gaps with less than or equal to 12 TBs) and one 'long' gap state (gaps with more than 12 TBs) and the corresponding transition probabilities are the conditional probabilities $p_{sl}$, $p_{ls}$, $p_{ll} = 1 - p_{ls}$ and $p_{ss} = 1 - p_{sl}$ as shown in **Figure D.6**. After entering a certain state, first the gap length is calculated via a Weibull distributed random number for either the small or the large gap lengths according to $P^{gap}_{short}$ and depending on the current upper layer state. Then, while staying in the same state of the upper layer Markov chain, a burst is calculated with different probabilities of either creating a Weibull distributed burstlength or a peak at short or long deterministic burstlength, like in the one-layer Karner model, again depending on the present upper layer state. After that the state of the upper layer Markov chain is changed and the procedure begins again with calculating the gap length.

Thus, via the upper layer Markov chain, the model introduces correlation between subsequent gaps and also between a gap and the following burst by using 11 parameters.

The one-layer Karner model is fully described by nine parameters ($a^{gap}_{short}$, $b^{gap}_{short}$, $a^{gap}_{long}$, $b^{gap}_{long}$, $P^{gap}_{short}$, $a^{burst}$, $b^{burts}$, $P^{burst}_{short}$, and $P^{burst}_{long}$) and assumes uncorrelated both gaps and bursts. The two-layer Karner model requires 11 parameters and allows for modeling of the bursts and gaps correlation as well. These parameters are: two upper layer Markov chain transition probabilities $p_{ss} = 1 - p_{sl}$ and $p_{ls} = 1 - p_{ll}$, for each of the correct states, two Weibull distribution parameters $a$ and $b$ are necessary, and two more Weibull distribution parameter for the erroneous states are necessary (equal for both upper layer Markov chain states). Furthermore, the probability of generating Weibull-distributed burst lengths in

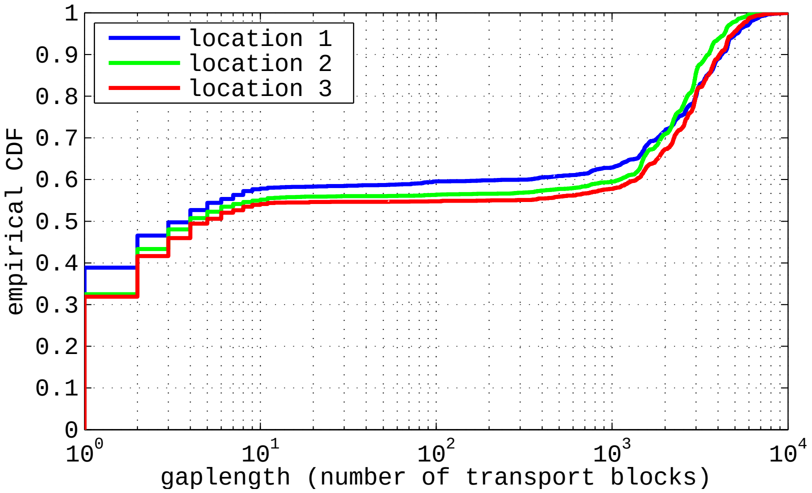


**Figure D.7**: Statistics of gap lengths based on measurements for static scenario.

both upper layer Markov chain states and of generating the deterministic burst lengths is necessary.

Note that both Karner models do not take into account radio bearer (RB) switching, that can be optionally configured in a UMTS network. A model for RB switching can be found in [147].

### D.2.3 Performance Comparison

In **Figure D.7** the empirical CDFs of the gap lengths (the number of error free transport blocks between two erroneous TBs), measured at three different locations, are shown. The measurements performed at three different locations show the same characteristic and thus, provide the possibility for a statistical modeling. Two main regions of occurrence of gap lengths can be observed: the short gaps with length smaller than 12 TBs and the long gaps with length greater than 500 TBs. Between these two main parts there is only a negligibly small probability of having a gap. The comparison of the burstlengths and gap lengths statistics for measured traces, memoryless, simplified Gilbert, and two-layer Karner model are shown in **Figure D.8**. The parameters of all tested models were obtained by least-squares fitting to the measured traces. The two-layer Karner model performs clearly better than the simplified Gilbert and memoryless error models. It corresponds well to the measured traces. Note that the measured traces used for comparison were different (measured at other location, at other time but in the same static scenario) from those used to design the Karner models. There is no real difference between the burstlength and the gap length on the one-layer and the two-layer Karner model, thus only the two-layer model was plotted here. The difference in the performance of these models was studied in detail in [150].

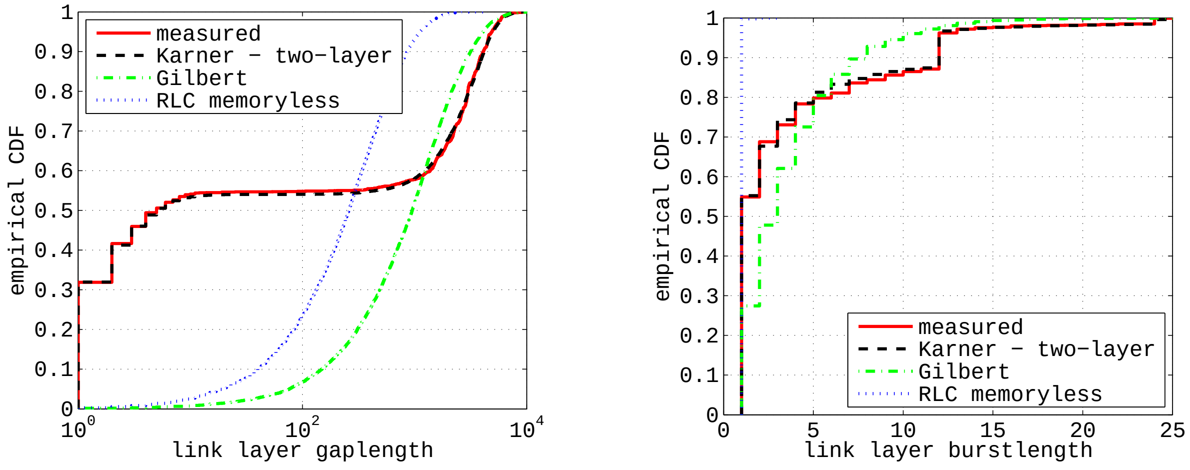
The choice of an appropriate link layer model has a great impact on the end-to-end performance of a video stream transmitted over such channel. This has been clearly demonstrated in [79] and is summarized in **Figure D.9**. There is only a small difference between the one-layer and the two-layer Karner model from the point of view of Y-PSNR (the correlation coefficient for one-layer Karner and measured trace is 0.9953, for two-layer Karned and measured trace it is 0.9964). Note that the mean link layer error probability was kept the same for all compared link layer error models. **Figure D.9** was obtained by means of experiments using JM H.264/AVC. For the experiments the ‘SoccerMatch’ video sequence with QVGA picture resolution and a frame rate of 10, containing a soccer match with different scenes (for more details see Appendix B.2.5). The sequence was encoded using I and P frames only, every 40th frame being an I frame. The slicing mode two with maximum 750 bytes per slice was chosen. The quantization parameter was set to QP = 26, the RDO was disabled. To obtain reliable results, the video was decoded several times resulting in approximately 10 hours of video streaming.

## D.3 Transport Layer Error Model

THE IP protocol forms the transport layer in UMTS packet switched transmission of video streaming. In practice, the mostly used IP layer error model is the memoryless error model. In this thesis (cf. Chapter 3 and 4), however, an error model obtained from the link layer error traces is used, since it better reflects the reality.

### D.3.1 Memoryless Error Model

The IP packets have typically different but limited lengths. The IP Maximum Transmission Unit (MTU) is the largest size of IP packet data which may be transferred using a specific data link connection. The typical value for MTU is 1500 bytes. The memoryless error model is fully defined by the *packetloss* ratio, the errors are temporally uncorrelated. It disregards the varying lengths of the IP packets, i.e. the packetloss ratio is constant over the duration of the stream transmission independently of the varying IP packet lengths. In fact, such memoryless error model corresponds to BSC for the case where erroneous packets are represented by ones and error-free packets by zeros.



(a) gaplengths (b) burstlenghts

**Figure D.8**: Comparison of the link layer statistic of the gap lengths and burstlengths.

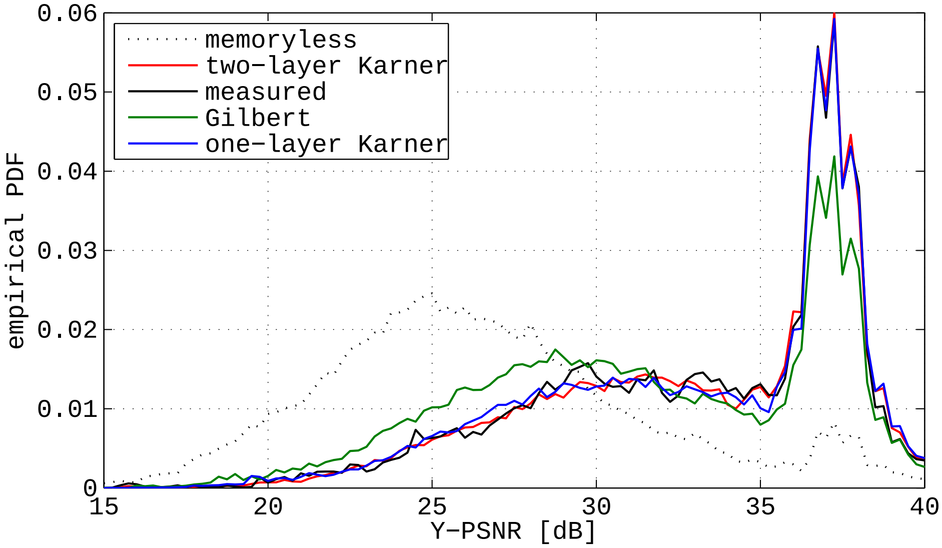


**Figure D.9**: Distribution of the frame Y-PSNR for the various link layer error models.

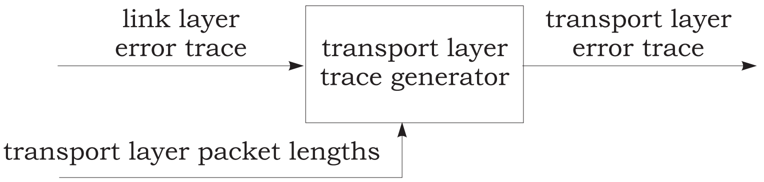


**Figure D.10**: Generation of a transport layer error trace.

### D.3.2 Transport Layer Error Trace

Transport layer error traces used in this thesis were obtained by mapping the link layer error traces onto the transport layer packet lengths as shown in **Figure D.10**. The transport layer packet lengths were obtained from the H.264/AVC encoder or decoder. Surprisingly, the impact of both the memoryless IP error model and the IP layer trace on the frame Y-PSNR distribution was shown to be very similar [79] for the typical error handling. A notable difference in Y-PSNR was observed for the video stream encoded without intra-predicted frames. This is because the resulting IP gap length and burstlength distributions for the memoryless model and the measured traces differ substantially as shown in **Figure D.11**. Thus, in this thesis the transport layer error traces are a preferred approach.

## D.4 Prediction of Error Probability

BASED on the Karner model for static scenario of UMTS DCH, the probability of error can be estimated.

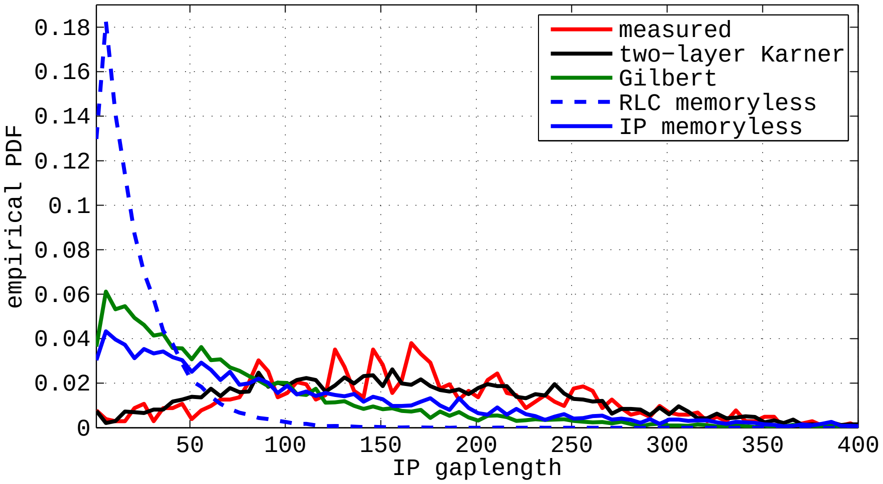


**Figure D.11**: Comparison of IP gap lengths for the various link and transport layer error models.

Let an erroneously received TB be indicated by "1" while "0" means an error-free transmission. A positive integer in the exponent determines the number of consecutive erroneous or error-free TBs. For example, the sequence "000001" can be written as "$0^5 1$". A gap with length $m$ (a non-negative integer) is defined as the number of zeros between two ones. The PMF (Probability Mass Function) of the gap lengths $m$ is given by

$$p_M(m) = \mathrm{P}\{M = m\} \triangleq P(0^m 1|1). \tag{D.3}$$

The conditional probability $P(B|A)$ means the probability of sequence $B$, following sequence $A$. By definition,

$$\sum_{m=0}^{\infty} P(0^m 1|1) = 1 \tag{D.4}$$

and $P(0^0 1|1) = 0$, as gaps with length zero are not considered for being gap lengths. The CDF of the gap lengths is then defined as

$$F_M(m) = \mathrm{P}\{M \leq m\} = \sum_{k=0}^{m} P(0^k 1|1). \tag{D.5}$$

The conditional link error probability $P(1|10^m)$ (the error probability conditioned on the number of error-free TBs since the last error) corresponds to the expected failure rate [152], and can be expressed by

$$P(1|10^m) = \frac{P(0^m 1|1)}{P(0^m|1)} \approx \frac{p_M(m)}{1 - F_M(m)}, \tag{D.6}$$

where $P(0^m|1)$ corresponds approximately to the CCDF (Complementary Cumulative Distribution Function) of the gap lengths:

$$P(0^m|1) = \sum_{k=m}^{\infty} P(0^k 1|1) = 1 - F_M(m) + p_M(m) \approx 1 - F_M(m). \tag{D.7}$$

$P(0^m|1)$ denotes the probability of having a gap length of at least length $m$.

The expected failure rate $P(1|10^m)$ can be estimated based on the feedback information (e.g. RLC acknowledgements) at the transmitter. However, the feedback information is not available immediately, but after the feedback delay $d_{\mathrm{FB}}$ after the transmission of each TB. Since the size of the short gaps in the Karner model is about the size of $d_{\mathrm{FB}}$, only the statistics of the long gaps can be used for the error prediction. The gap length $M$ of the long gaps obeys a discrete Weibull distribution given by the following PMF

$$p_M^{(Weibull)}(m) = ba^{-b}m^{b-1}e^{-(\frac{m}{a})^b} \tag{D.8}$$

with scale parameter $a \in \mathbb{R}, a > 0$ and shape parameter $b \in \mathbb{R}, b > 0$. After inserting the Weibull PMF for $P(0^m1|1)$ and the Weibull CCDF for $P(0^m|1)$ in (D.6), the estimated failure rate is

$$\widehat{P}(1|10^m)_{Weibull} = \frac{ba^{-b}m^{b-1}e^{-(\frac{m}{a})^b}}{e^{-(\frac{m}{a})^b}} = ba^{-b}m^{b-1}. \tag{D.9}$$

The conditional link error probability $P(1|10^m)$ can be estimated by $b = 2.0176$, corresponding to the statistics of the long gaps obtained by measurements (see Appendix D.2.2) in a static scenario. With $b \approx 2$, the Weibull distribution becomes Rayleigh and the estimated expected failure ratio can be expressed by the much simpler term

$$\widehat{P}(1|10^m) = \frac{2m}{a^2}, \tag{D.10}$$

with $a = 3350$. Thus, it is linear increasing with the error-free run length $m$ (the number of error-free TBs received so far after the last error).

# Bibliography

[1] H. Holma, A. Toscala, *WCDMA for UMTS: radio Access For Third Generation Mobile Communication,* Third Edition, John Wiley & Sons, Ltd., 2004.

[2] 3rd Generation Partnership Project, website: `http://www.3gpp.org`.

[3] ITU-T H.263, Series H: Audiovisual and multimedia systems, Infrastructure of audiovisual services — coding of moving video, "Video coding for low bit rate communication," Jan. 2005.

[4] ISO/IEC–14496-2, "Coding of Audio-Visual Objects," part 2: visual, 2001.

[5] ITU-T H.264, Series H: Audiovisual and multimedia systems, Infrastructure of audiovisual services — coding of moving video, "Advanced video coding for generic audiovisual services," Mar. 2005.

[6] D. Marpe, T. Wiegand, G.J. Sullivan, "The H.264/MPEG4 Advanced Video Coding Standard and its Application," IEEE Communications Magazine, vol. 44, no. 8, pp. 134–143, Aug. 2006.

[7] I.E.G. Richardson, *H.264 and MPEG-4 Video Compression, Video Coding for the Next-Generation Multimedia,* John Wiley & Sons Ltd., Mar. 2005.

[8] ITU-R BT.601-6, Series BT: Broadcast service (television), "Studio encoding parameters of digital television for standard 4:3 and wide screen 16:9 aspect ratios," Jan. 2007.

[9] S. Winkler, *Digital Video Quality,* JohnWiley & Sons, Chichester, 2005.

[10] P. List, A. Joch, J. Lainema, G. Bjontegaard, M. Karczewicz, "Adaptive Deblocking Filter," IEEE Trans. on Circuits and Systems for Video Technology, vol. 13, no. 7, pp. 614–619, Jul. 2003.

[11] S. Wenger et al., "RTP Payload Format for H.264 Video," IETF RFC 3984, Feb. 2005.

[12] H.264/AVC Software Coordination, "Joint Model Software", ver. 11.0, available in `http://iphome.hhi.de/suehring/tml/`.

[13] J.R. Corbera, P.A. Chou, S.L. Regunathan, "A Generalized Hypothetical Reference Decoder for H.264/AVC," IEEE Trans. on Circuits and Systems for Video Technology, vol. 13, no. 7, pp. 574–587, Jul. 2003.

[14] S. Wenger, "H.264/AVC over IP," IEEE Trans. on Circuits and Systems for Video Technology, vol. 13, no. 7, pp. 645–657, Jul. 2003.

[15] S. Branguolo, N. Tizon, B.P. Popescu, B. Lehembre, "Video Transmission over UMTS Networks Using UDP/IP," in Proc. of 14th European Signal Processing Conference (EUSIPCO), Florence, Italy, Sep. 2006.

[16] T. Wiegand, G.J. Sullivan, G. Bjontegaard, A. Luthra, "Overview of the H.264/AVC Video Coding Standard," IEEE Trans. on Circuits and Systems for Video Technology, vol. 13, no. 7, pp. 560–576, Jul. 2003.

[17] J. Ostermann, J. Bormans, P. List, D. Marpe, M. Narroschke, F. Pereira, T. Stockhammer, T. Wedi, "Video Coding with H.264/AVC: Tools, Performance and Complexity," IEEE Mag. on Circuits and Systems, vol. 4, no. 1, pp. 7–28, 2004.

[18] T. Stockhammer, M.M. Hannuksela, "H.264/AVC Video for Wireless Transmission," IEEE Mag. on Wireless Communications, Special Issue: Advances in Wireless Video, vol. 12, no. 4, pp. 6–13, Aug. 2005.

[19] H. Schwarz, D. Marpe, T. Wiegand, “Overview of the scalable H.264/MPEG4-AVC extension,” in Proc. of IEEE Int. Conf. on Image Processing (ICIP), pp. 161–164, Oct. 2006.

[20] G. Liebl, T. Schierl, T. Wiegand, T. Stockhammer, “Advanced wireless multiuser video streaming using the scalable video coding extensions of H.264/MPEG4-AVC,” in Proc. of IEEE Int. Conf. on Multimedia and Expo (ICME), pp. 625–628, Jul. 2006.

[21] V.K. Goyal, ”Multiple Description Coding: Compression Meets the Network,” IEEE Signal Processing Mag., vol. 18, no. 5, pp. 74–93, Sep. 2001.

[22] N. Franchi, M. Fumagalli, R. Lancini, S. Tubaro, “Multiple Description Video Coding for Scalable and Robust Transmission Over IP,” IEEE Trans. on Circuits and Systems for Video Technology, vol. 15, no. 3, pp. 321–334, Mar. 2005.

[23] I. Radulovic, P. Frossard, O. Verscheure, ”Adaptive Video Streaming In Lossy Networks: Versions or Layers?,” in Proc. of IEEE Int. Conf. on Multimedia and Expo (ICME), Taipei, Taiwan, Jun. 2004.

[24] J. Chakareski, S. Han, B. Girod, “Layered coding vs. multiple descriptions for video streaming over multiple paths,” Multimedia Systems, Springer, online journal publikation, DOI: 10.1007/s00530-004-0162-3, Jan. 2005.

[25] B.H. Pathac, G. Childs, M. Ali, “UEP implementation for MPEG-4 video quality improvement on RLC layer of UMTS,” IEE Electronic Letters, vol. 41, no. 13, DOI: 10.1049/el:20051314, Jun. 2005.

[26] J. Klaue, J. Gross, H. Karl, A. Wolisz, “Semantic-aware Link Layer Scheduling of MPEG-4 Video Streams in Wireless Systems,” in Proc. of. Applications and Services in Wireless Networks (AWSN), Bern, Switzerland, Jul. 2003.

[27] Q. Qu, Y. Pei, J.W. Modestino, “An Adaptive Motion-Based Unequal Error Protection Approach for Real-Time Video Transport Over Wireless IP Networks,” IEEE Trans. on Multimedia, vol. 8, no. 5, pp. 1033–1044, Oct. 2006.

[28] N. Thomos, S. Argyropoulos, N.V. Boulgouris, M.G. Strintzis, “Error-Resilient Transmission of H.264/AVC Streams Using Flexible Macroblock Ordering,” in Proc. of 2nd European Workshop on the Integration of Knowledge, Semantic, and Digital Media Techniques (EWIMT), London, UK, Nov. 2005.

[29] P. Bucciol, E. Masala, J.C. De Martin, “Perceptual ARQ for H.264 Video Streaming over 3G Wireless Networks,” in Proc. of Int. Conf. on Communications (ICC), vol. 3, pp. 1288–1292, Paris, France, Jun. 2004.

[30] C.E. Shannon, “A Mathematical Theory of Communication,” Bell System Technical Journal, vol. 27, pp. 379–423, Jul. 1948.

[31] R.E. van Dyck, D.J. Miller, “Transport of wireless video using separate, concatenated, and joint source-channel coding,” IEEE Proceedings, vol. 87, no. 10, pp. 1734–1750, Oct. 1999.

[32] N. Farvardin, “A Study of Vector Quantization for Noisy Channels,” IEEE Trans. on Information Theory, vol. 36, no. 4, pp. 799–809, Jul. 1990.

[33] K. Sayood, H.H. Otu, N. Demir, “Joint Source/Channel Coding for Variable Length Codes,” IEEE Trans. on Communications, vol. 48, no. 5, May 2000.

[34] H. Nguyen, P. Duhamel, “Robust Source Decoding of Variable-Length Encoded Video Data Taking into Account Source Constraints,” IEEE Trans. on Communications, vol. 53, no. 7, pp. 1077–1084, Jul. 2005.

[35] C. Lee, M. Kieffer, P. Duhamel, “Robust Reconstruction of Motion Vectors Using Frame Expansion,” in Proc. of IEEE Int. Conf. on Acoustic, Speech, and Signal Processing (ICASSP), pp. 617–620, May 2004.

[36] Y. Takishima, M. Wada, H. Murakami, ”Reversible Variable Length Codes,”IEEE Trans. on Communicatios, vol. 42, no. 2/3/4, 1994.

[37] J. Wen, J.D. Villasenor, “Reversible Variable Length Codes for Efficient and Robust Image and Video Coding,” in Proc. of IEEE Data Compression Conference, Snowbird, Utah, USA, Apr. 1998.

[38] G. Côté, S. Shirani, F. Kossentini, “Optimal Mode Selection and Synchronization for Robust Video Communications over Error-Prone Networks,” IEEE Jour. on Selected Areas in Communications, vol. 18, no. 6, pp. 952–965, Jun. 2000.

[39] M. Chiani, M.G. Martini, “On Sequential Frame Synchronization in AWGN Channels,” IEEE Trans. on Communications, vol. 54, no. 2, pp. 339–348, Feb. 2006.

[40] Z. He, S.K. Mitra, “From Rate-Distortion Analysis to Resource-Distortion Analysis,” IEEE Circuits and Systems Magazine, vol. 5, no. 3, pp. 6–18, Mar. 2005.

[41] M. Kalman, P. Ramanathan, B. Girod, “Rate-distortion Optimized Video Streaming with Multiple Deadlines,” in Proc. of IEEE Int. Conf. in Image Processing (ICIP), Sep. 2003.

[42] M. Kalman, B. Girod, “Rate-Distortion Optimized Video Streaming Using Conditional Packet delay Distributions,” IEEE Workshop on Multimedia Signal Processing, Siena, Italy, Sep. 2004.

[43] P. Chou, Z. Miao, “Rate-Distortion Optimized Streaming of Packetized Media,” IEEE Trans. on Multimedia, vol. 8, no. 2, pp. 390–404, Apr. 2006.

[44] H. Zimmermann, “OSI Reference Model — The ISO Model of Architecture for Open Systems Interconnection,” IEEE Trans. on Communications, vol. 28, no. 4, pp. 425–432, Apr. 1980.

[45] M. van der Schaar, S. Shankar, “Cross-Layer Multimedia Transmission: Challenges, Principles, and New Paradigms,” IEEE Mag. on Wireless Communications, Special Issue: Advances in Wireless Video, vol. 12, no. 4, pp. 6–13, Aug. 2005.

[46] J. Chen, T. Lv, H. Zheng, “Joint Cross-Layer Design for Wireless QoS Content Delivery,” EURASIP Journal on Applied Signal Processing, vol. 2, pp. 167-182, 2005.

[47] L.U. Choi, M.T. Ivrlac, E. Steinbach, J.A. Nossek, “Bottom-Up Approach to Cross-layer Design for Video Transmission ove Wireless Channels,” IEEE 61th Vehicular Technology Conference (VTC), vol. 5, pp. 3019–3023, Stockholm, Sweden, May 2005.

[48] M.T. Sun, A.R. Reibman, ed., “Chapter 3: Error Concealment,” *Compressed Video over Networks,* Signal Processing and Communications Series, Marcel Dekker Inc., New York, pp. 217-250, 2001.

[49] Y. Xu, Y. Zhou, “H.264 Video Communication Based Refined Error Concealment Schemes,” IEEE Trans. on Consumer Electronics, vol. 50, no. 4, pp. 1135-1141, Nov. 2004.

[50] D.J.C. MacKay, *Information Theory, Inference, and Learning Algorithms,* Cambridge University Press, fifth printing, 2006.

[51] O. Nemethova, M. Ries, E. Siffel, M. Rupp, “Subjective Evaluation of Video Quality for H.264 Encoded Sequences,” in Proc. of Joint IST Workshop on Mobile Future & Symp. on Trends in Communications (SympoTIC), pp. 191–194, Bratislava, Slovakia, pp. 191–194, Oct. 2004.

[52] O. Nemethova, M. Ries, E. Siffel, M. Rupp, “Quality Assessment for H.264 Coded Low-Rate and low-Resolution Video Sequences,” in Proc. of IASTED Communications, Internet and Information technology (CIIT), St.Thomas, US Virgin Islands, Nov. 2004.

[53] M. Ries, R. Puglia, T. Tebaldi, O. Nemethova, M. Rupp, “Audivisual Quality Estimation for Mobile Streaming Services,” in Proc. of 2nd Int. Symp. on Wireless Communications (ISWCS), Siena, Italy, pp. 173–177, Sep. 2005.

[54] O. Nemethova, M. Ries, A. Dantcheva, S. Fikar, M. Rupp, ”Test Equipment of Time-Variant Subjective Perceptual Video Quality in Mobile Terminals,” in Proc. of Int. Conf. on Human Computer Interaction (HCI), pp. 1–6, Phoenix, USA, Nov. 2005.

[55] M. Ries, O. Nemethova, M. Rupp, ”Reference-Free Video Quality Metric for Mobile Streaming Applications,” in Proc. of the Joint Int. Symp. on DSP and Communications Systems and Workshop on the Internet, Telecommunications and Signal Processing (DSPCS&WITSP), Sunshine Coast, Australia, Dec. 2005.

[56] O. Nemethova, M. Ries, M. Zavodsky, M. Rupp, “PSNR-Based Estimation of Subjective Time-Variant Video Quality for Mobiles,” in Proc. of 5th Int. Conf. on Measurement of Audio and Video Quality in Networks (MESAQIN), Prague, Czech Republic, Jun. 2006.

[57] M. Ries, O. Nemethova, M. Rupp, “Motion Based Reference-Free Quality Estimation for H.264/AVC Video Streaming,” in Proc. of IEEE Int. Symp. on Wireless Pervasive Computing (ISWPC), San Juan, Puerto Rico, US, Feb. 2007.

[58] M. Ries, O. Nemethova, M. Rupp, “Reference-free video quality estimator for video streaming,” submitted for Austrian patent 22.12.2006, No. PCT/AT2006/000539.

[59] M. Ries, O. Nemethova, C. Crespi, M. Rupp, “Content Based Video Quality Estimation for H.264/AVC Video Streaming,” in Proc. of IEEE Wireless Communications and Networking Conf. (WCNC), Hong Kong, pp. 2668–2673, Mar. 2007.

[60] M. Ries, O. Nemethova, M. Rupp, “Performance evaluation of mobile video quality estimators,” invited paper, accepted to 15th European Signal Processing Conference (EUSIPCO), Poznan, Polen, Sep. 2007.

[61] M. Barni, F. Bartolini, P. Bianco, ”On the performance of syntax-based error detection in H.263 video coding: a quantitative analysis,” Electronic Imaging 2000 – SPIE Conf. on Image and Video Communications, San Jose, CA, Jan. 2000.

[62] L. Superiori, O. Nemethova, M. Rupp, “Performance of a H.264/AVC Error Detection Algorithm Based on Syntax Analysis,” 4th Int. Conf. on Mobile Computing and Multimedia (MoMM), Yogyakarta, Indonesia, pp. 49–58, Dec. 2006.

[63] L. Superiori, O. Nemethova, M. Rupp, “Detection of Visual Impairment in the Pixel Domain of Corrupted H.264/AVC Packets,” submitted to Int. Picture Coding Symposium, Lisboa, Portugal, Nov. 2007.

[64] O. Nemethova, J. Canadas, M. Rupp, “Improved Detection for H.264 Encoded Video Sequences over Mobile Networks,” in Proc. of Int. Symp. on Communication Theory and Applications (ISCTA), Ambleside, UK, July 2005.

[65] O. Nemethova, G.C. Forte, M. Rupp, “Robust Error Detection for H.264/AVC Using Relation Based Fragile Watermarking,” in Proc. of 13th Int. Conf. on Systems, Signals and Image Processing, Budapest, Hungary, Sep. 2006.

[66] M. Chen, Y. He, R.L. Lagendijk, ”A Fragile Watermark Error Detection Scheme fot Wireless Video Communications,” IEEE Trans. on Multimedia, vol. 7, no. 2, pp. 201-211, Apr. 2005.

[67] C. Bergeron, C. Lamy-Bergot, “Soft-input decoding of variable-length codes applied to the H.264 standard,” in Proc. of IEEE Workshop on Multimedia Signal Processing (MMSP), Siena, Italy, Sep. 2004.

[68] C. Weidmann, P. Kadlec, O. Nemethova, A. Al-Moghrabi, “Combined Sequential Decoding and Error Concealment of H.264 Video,” in Proc. of IEEE Workshop on Multimedia Signal Processing (MMSP), Siena, Italy, pp. 299–302, Sep. 2004.

[69] C. Weidmann, O. Nemethova, “Improved Sequential Decoding of H.264 Video with VLC Resynchronization,” 15th IST Mobile & Wireless Communications Summit, Myconos, Greece, Jun. 2006.

[70] O. Nemethova, A. Al-Moghrabi, M. Rupp, “Flexible Error Concealment for H.264 Based on Directional Interpolation,” Proc. of Int. Conf. on Wireless Networks, Communications and Mobile Computing (WirelessCom), Maui, Hawaii, pp. 1255–1260, June 2005.

[71] A. Dimou, O. Nemethova, M. Rupp, “Scene Change Detection for H.264 Using Dynamic Threshold Techniques,” Proc. of 5th EURASIP Conf. on Speech and Image Processing, Smolenice, Slovakia, June 2005.

[72] O. Nemethova, A. Al-Moghrabi, M. Rupp, “An Adaptive Error Concealment Mechanism for H.264 Encoded Low-Resolution Video Streaming,” 14th European Signal Processing Conference (EUSIPCO), Florence, Italy, Sep. 2006.

[73] O. Nemethova, W. Karner, A. Al-Moghrabi, M. Rupp, “Cross-Layer Error Detection for H.264 Video over UMTS,” in Proc. of Wireless Personal Mobile Communications, International Wireless Summit, Aalborg, Denmark, Sep. 2005.

[74] W. Karner, O. Nemethova, M. Rupp, “Link Error Prediction in Wireless Communication Systems with Quality Based Power Control,” accepted for IEEE Int. Conf. on Communications (ICC), Glasgow, UK, June 2007.

[75] O. Nemethova, W. Karner, M. Rupp, “Error Prediction Based Redundancy Control for Robust Transmission of Video over Wireless Links,” accepted for IEEE Int. Conf. on Communications (ICC), Glasgow, UK, June 2007.

[76] W. Karner, O. Nemethova, M. Rupp, “Verfahren und Vorrichtung zum Steuern der paketweisen bertragung von Daten,” Patent: Austria, Nr. 905/2006, submitted: 24.05.2006.

[77] W. Karner, O. Nemethova, M. Rupp, “Link Error Prediction Based Cross-Layer Scheduling for Video Streaming over UMTS,” in Proc. of 15th IST Mobile & Wireless Communications Summit, Myconos, Greece, Jun. 2006.

[78] O. Nemethova, W. Karner, C. Weidmann, M. Rupp, “Distortion-minimizing network-aware scheduling for UMTS video streaming,” invited paper, accepted to 15th European Signal Processing Conference (EUSIPCO), Poznan, Polen, Sep. 2007.

[79] W. Karner, O. Nemethova, M. Rupp, “The Impact of Link Error Modeling on the Quality of Streamed Video in Wireless Networks,” in Proc. of 3rd Int. Symp. on Wireless Communication Systems (ISWCS), Valencia, Spain, Sep. 2006.

[80] 3GPP TSG TR 26.937, “Transparent end-to-end packet switched streaming service (PSS); Real-time Transport Protocol (RTP) usage model,” ver. 5.0.0, Aug. 2003.

[81] I.J. Cox, J. Kilian, F.T. Leighton, T. Shamoon, “Secure spread spectrum watermarking for multimedia,” IEEE Transactions on Image Processing, vol. 6, no. 12, pp. 1673–1687, Dec. 1997.

[82] F. Bellifemine et al., ”Statistics analysis of the 2D-DCT coefficients of the differential signal of images,” Signal Processing: Image Communications, vol. 4, 1992.

[83] A. Kiely, S. Dolinar, M. Klimesh, A. Matache, ”Error Containment in Compressed Data Using Sync Markers,” Proc. of Int. Conf. on Information Theory (ICIT), Jun. 2000.

[84] K. Sayood, *Data Compression,* Second Edition, Morgan Kaufmann Publishers, 2000.

[85] S.W. Golomb, “Run-Length Encoding,” IEEE Trans. on Information Theory (Corresp.), vol. 12, pp. 399–401, July 1966.

[86] P. Elias, “Universal Codeword Sets and Representations of the Integers,” IEEE Trans. on Inf. Theory, vol. 21, no. 2, pp. 194–203, Mar. 1975.

[87] R.M. Fano, “A heuristic discussion of probabilistic decoding,” IEEE Trans. on Information Theory, vol. 9, pp. 64-73, Apr. 1963.

[88] J.L. Massey, “Variable-length codes and the Fano metric,” IEEE Trans. on Information Theory, vol. 18, pp. 196-198, Jan. 1972.

[89] J.W. Suh, Y.S. Ho, “Error Concealment based on Directional Interpolation,” IEEE Trans. on Consumer Electronics, vol. 43, no. 3, pp. 295–302, Aug. 1997.

[90] W. Kwok, H. Sun, “Multi-directional Interpolation For Spatial Error Concealment,” in Proc. of IEEE Int. Conf. on Consumer Electronics (ICCE), pp. 220–221, Jun. 1993.

[91] M.C. Hong, L. Kondi, H. Schwab, A.K. Katsaggelos, “Error Concealment Algorithms for Compressed Video,” Signal Processing: Image Communications, special issue on Error Resilient Video, vol. 14, no. 6–8, pp. 437–492, 1999.

[92] A. Koschan, M. Abidi, “Detection and Classification of Edges in Color Images,” IEEE Signal Processing Magazine, vol. 22, no. 1, pp. 64–73, Jan. 2005.

[93] W.Y. Kung, C.S. Kim, C.J. Kuo, “A Spatial-Domain Error Concealment Method With Edge Recovery And Selective Directional Interpolation,” in Proc. of IEEE Int. Conf. on Acoustic, Speech, and Signal Processing (ICASSP), Hong Kong, 2003.

[94] Y. Wang, Q.F. Zhu, L. Shaw, “Maximally Smooth Image Recovery in Transform Coding,” IEEE Trans. on Communiacations, vol. 41, no. 10, pp. 1544–1551, Oct. 1993.

[95] Q.F. Zhu, Y. Wang, L. Shaw, “Coding and Cell-loss Recovery in DCT-Based Packet Video,” IEEE Trans. on Circuits and Systems for Video Technology, vol. 3, no. 3, pp. 248–258, Jun. 1993.

[96] W. Zhu, Y. Wang, Q.F. Zhu, “Second-Order Derivative-Based Smoothness Measure for Error Concealment in DCT-Based Codecs,” IEEE Trans. on Circuits and Systems for Video Technology, vol. 8, no. 6, pp. 713–718, Oct. 1998.

[97] E. Hinterberger, “Methoden zur Datenrueckgewinnug bei Bild- und Videouebertragung,” report of a research project supervised by O. Nemethova and M. Rupp, available at `www.nt.tuwien.ac.at`, Vienna University of Technology, Feb. 2006.

[98] H. Sun, W. Kwok, ”Concealment of Damaged Block Transform Coded Images Using Projections onto Convex Sets,” IEEE Trans. on Image Processing, vol. 4, no. 4, pp. 470–477, Apr. 1995.

[99] D.C. Youla, H. Webb, “Image Restoration by the Method of Convex Projections: Part I — Theory,” IEEE Trans. on Medical Imaging, vol. M1-1, pp. 81–94, Oct. 1982.

[100] A.V. Oppenheim, A.S. Willsky, S.H. Nawab, *Signal & Systems,* second edition, Prentice Hall, 1996.

[101] J.W. Suh, Y.S. Ho, “Error Concealment Techniques for Digital TV,” IEEE Trans. on Broadcasting, vol. 48, no. 4, pp. 299–306, Dec. 2002.

[102] M.J. Chen, L.G. Chen, R.M. Weng, “Error Concealment of Lost Motion Vectors with Overlapped Motion Compensation,” IEEE Trans. on Circuits and Systems for Video Technology, vol. 7, no. 3, pp. 560–563, Jun. 1997.

[103] F.J. Harris, “On the Use of Windows for Harmonic Analysis with the Discrete Fourier Transform,” Proceedings of IEEE, vol. 66, no. 1, pp. 51–83, Jan. 1978.

[104] M. Turk, A. Pentland, “Eigenfaces for Recognition,” MIT Journal of Cognitive Neuroscience, vol. 3, no. 1, pp. 71–86, 1991.

[105] T.P.Ch. Chen, T. Chen, “Second-Generation Error Concealment for Video Transport over Error-Prone Channels,” Wiley Wireless Communications and Mobile Computing, vol. 2, no. 6, pp. 607–624, 2002.

[106] L. Superiori, O. Nemethova, M. Rupp, “Clustering-based Object Detection for Low-resolution Video Streaming,” IEEE Int. Symp. on Broadband Multimedia Systems and Broadcasting, Orlando, Florida, USA, Mar. 2007.

[107] S. Belfiore, M. Grangetto, G. Olmo, ”An error concealment algorithm for streaming video,” Proc. of IEEE Int. Conf. on Image Processing, vol. 3, pp. 649–652, Sep. 2003.

[108] S. Cen, P. Cosman, F. Azadegan, “Decision Trees for Error Concealment in Video Decoding,” IEEE Trans. on Multimedia, vol. 5, no. 1, pp. 1–7, Mar. 2003.

[109] Y.K. Wang, M.M. Hannuksela, V. Varsa, A. Hourunranta, M. Gabbouj, “The Error Concealment Feature in H.26L Test Model,” in Proc. of IEEE Int. Conf. Image Processing (ICIP), New York, USA, Sep. 2002.

[110] A. Hanjalic, *Content-based analysis of digital video,* Kluwer Academic Publishers, 2004.

[111] L. Su, Y. Zhang, W. Gao, Q. Huang, Y. Lu, ”Improved Error Concealment Algorithms Based on H.264/AVC Non-normative Decoder,” Proc. of IEEE Int. Conf. on Multimedia and Expo, Taipei, Taiwan, Jun. 2004.

[112] C.L. Huang, B.Y. Liao, "A Robust Scene-Change Detection Method for Video Segmentation," IEEE Trans. on Circuits and Systems for Video Technology, vol. 11, no. 12, pp. 1281–1288, Dec. 2001.

[113] S. Youm, W. Kim, "Dynamic Threshold Method For Scene Change Detection," in Proc. of IEEE Int. Conf. on Multimedia and Expo (ICME), vol. 2, pp. 337–340, 2003.

[114] X. Wang, Z. Weng, "Scene Abrupt Change Detection, Canadian Conference on Electrical and Computer Engineering," vol. 2, pp. 880–883, 2000.

[115] K.W. Sze, K.M. Lam, G. Qiu, "Scene Cut Detection Using The Colored Pattern Appearance Model," in Proc. of IEEE Int. Conf. on Image Processing (ICIP), Barcelona, Spain, vol. 2, pp. 1017–1020, Sep. 2003.

[116] H.C. Liu, G. Zick, "Automatic Determination of Scene Changes in MPEG Compressed Video," in Proc. of IEEE Int. Symp. on Circuits and Systems (ISCAS), Seattle, USA, vol. 1, pp. 764–767, Apr. 1995.

[117] H. Li, G. Liu, Z. Zhang, Y. Li, "Adaptive Scene-Detection Algorithm for VBR Video Stream," IEEE Trans. on Multimedia, vol. 6, no. 4, pp. 624–633, Aug. 2004.

[118] D.D. Feng, W.Ch. Sin, H.J. Zhang, *Multimedia Information Retrieval and Management: Technological Fundamentals and Applications,* Springer, 2003.

[119] J. Postel, "User Datagram Protocol," RFC 768, Aug. 1980.

[120] L.A. Larzon, M. Degermark, S. Pink, L.E. Johnsson, G. Fairhurst, "The Lightweight User Datagram Protocol (UDP-Lite)," RFC 3828, Jul. 2004.

[121] H. Zheng, J. Boyce, "An Improved UDP Protocol for Video Transmission Over Internet-to-Wireless Networks," IEEE Trans. on Multimedia, vol. 3, no. 3, pp. 356–365, Sep. 2001.

[122] H. Fattah, C. Leung, "An Overview of Scheduling Algorithms in Wireless Multimedia Networks," IEEE Wireless Communications, pp. 76–85, Oct. 2002.

[123] E. Cianca, F.H.P. Fitzek, M. DeSanctis, M. Bonanno, R. Prasad, M. Ruggieri, "Improving Performance for Streaming Video Services over CDMA-Based Wireless Networks," in Proc. of Int. Symp. on Wireless Personal Multimedia Communications (WPMC), Padova, Italy, Sep. 2004.

[124] S.H. Kang, A. Zakhor, "Packet Scheduling Algorithm for Wireless Video Streaming," in Proc. of 12th Intl. Packetvideo Workshop 2002 (PV), Pittsburgh PA, USA, 2002.

[125] R.S. Tupelly, J. Zhang, E.K.P. Chong, "Opportunistic Scheduling for Streaming Video in Wireless Networks," in Proc. of the Conf. on Information Sciences and Systems, Johns Hopkins University, Baltimore, MD, Mar. 2003.

[126] P. Koutsakis, "Scheduling and Call Admission Control for Burst-Error Wireless Channels," in Proc. of 10th IEEE Symp. on Computers and Communications (ISCC), La Manga del Mar Menor, Cartagena, Spain, Jun. 2005.

[127] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, W. Weiss, "An Architecture for Differentiated Services," RFC 2475, Dec. 1998.

[128] M. Kac, "On the Notion of Recurrence in Discrete Stochastic Processes," Bulletin of the American Mathematical Society 53, pp. 1002–1010, 1947.

[129] C. Bergeron, C. Lamy-Bergot, "Modelling H.264/AVC sensitivity for error protection in wireless transmissions," in Proc. of IEEE Workshop on Multimedia Signal Processing (MMSP), Oct. 2006.

[130] S. Kanumuri, P.C. Cosman, A.R. Reibman, V.A. Vayshampayan, "Modeling Packet-Loss Visibility in MPEG-2 Video," IEEE Trans. on Multimedia, vol. 8, no. 2, pp. 341–355, Apr. 2006.

[131] F. De Vito, D. Quaglia, J.C. De Martin, "Model-based Distortion Estimation For Perceptual Classification of Video Packets," in Proc. of IEEE Workshop on Multimedia Signal Processing (MMSP), Sep. 2004.

[132] K. Stuhlmüller, N. Färber, M. Link, B. Girod, "Analysis of Video Transmission over Lossy Channels," IEEE Jour. on Selected Areas in Communications, vol. 18, no. 6, pp. 1012–1032, Jun. 2000.

[133] Carlos Teijeiro Castella, "RLC Based Distortion Model for H.264 Video Streaming," Vienna University of Technology, Apr. 2006.

[134] ITU-T Recommendation P.910, "Subjective video quality assessment methods for multimedia applications," Sep. 1999.

[135] 3GPP TSG TS 25.401, "UTRAN Overall Description," ver. 4.6.0, Jan. 2003.

[136] 3GPP TSG TS 23.107, "Quality of Service (QoS) Concept and Architecture," ver. 4.6.0, Jan. 2003.

[137] 3GPP TSG TS 25.321, "Medium Access Control (MAC); Protocol Specification," ver. 4.10.0, Jun. 2004.

[138] 3GPP TSG TS 25.322, "Radio Link Control (RLC); Protocol Specification," ver. 4.12.0, Jun. 2004.

[139] 3GPP TSG TS 25.323, "Packet Data Convergence Protol (PDCP); Protocol Specification," ver. 4.6.0, Sep. 2002.

[140] 3GPP TSG TS 25.331, "Radia Resource Control (RRC); Protocol specification," ver. 4.17.0, Mar. 2005.

[141] 3GPP TSG TR 25.993, "Typical Examples of Radio Access Bearers (RABs) and Radio Bearers (RBs) Supported by Universal Terrestrial Radio Access (UTRA)," ver. 4.2.0, Oct. 10.

[142] 3GPP TSG TS 34.108, "Common Test Environments for User Equipment (UE) Conformance Testing," ver. 4.11.0, Jun. 2004.

[143] 3GPP TSG TS 25.212, "Multiplexing and channel codeing (FDD)," ver. 4.6.0, Sep. 2002.

[144] E.N. Gilbert, "Capacity of a burst-noise channel,"Bell Systems Technical Journal, vol. 39, pp. 1253–1265, Sep. 1960.

[145] Qi Qu, Y. Pei, J.W. Modestino, X. Tian, "Source-Adaptive FEC/UEP Coding For Video Transport Over Bursty Packet Loss 3G UMTS Networks: A Cross-Layer Approach," 60th IEEE Vehicular Technology Conference (VTC2004-Fall), vol. 5, pp. 3150–3154, Sep. 2004.

[146] E.O. Elliot, "Estimates of error rates for codes on burst-noise channels," Bell Systems Technical Journal, vol. 42, pp. 1977–1997, Sep. 1963.

[147] W. Karner, O. Nemethova, M. Rupp, "A Measurement Based Model for UMTS DL DCH Dynamic Bearer Type Switching," in Proc. of IEEE Int. Symp. on Wireless Pervasive Computing (ISWPC), Phuket, Thailand, Jan. 2006.

[148] TEMS product description, available at `http://www.ericsson.com/products/hp/TEMS_Products_pa.shtml`

[149] W. Karner, P. Svoboda, M. Rupp, "A UMTS DL DCH Error Model Based on Measurements in Live Networks," in Proc. of 12th Int. Conf. on Telecommunications (ICT), Capetown, South Africa, May 2005.

[150] W. Karner, M. Rupp, "Measurement based Analysis and Modeling of UMTS DCH Error Characteristics for Static Scenarios," in Proc. of 8th Int. Symp. on DSP and Communication Systems (DSPCS), Sunshine Coast, Australia, Dec. 2005.

[151] B.D. Fritchman, "A Binary Channel Characterization Using Partitioned Markov Chains," IEEE Trans. Information Theory, vol. 13, no. 2, pp. 221-227, Apr. 1967.

[152] A. Papoulis, P.S. Unnikrishna, *Probability, random variables, and stochastic processes,* McGraw-Hill, 2002.